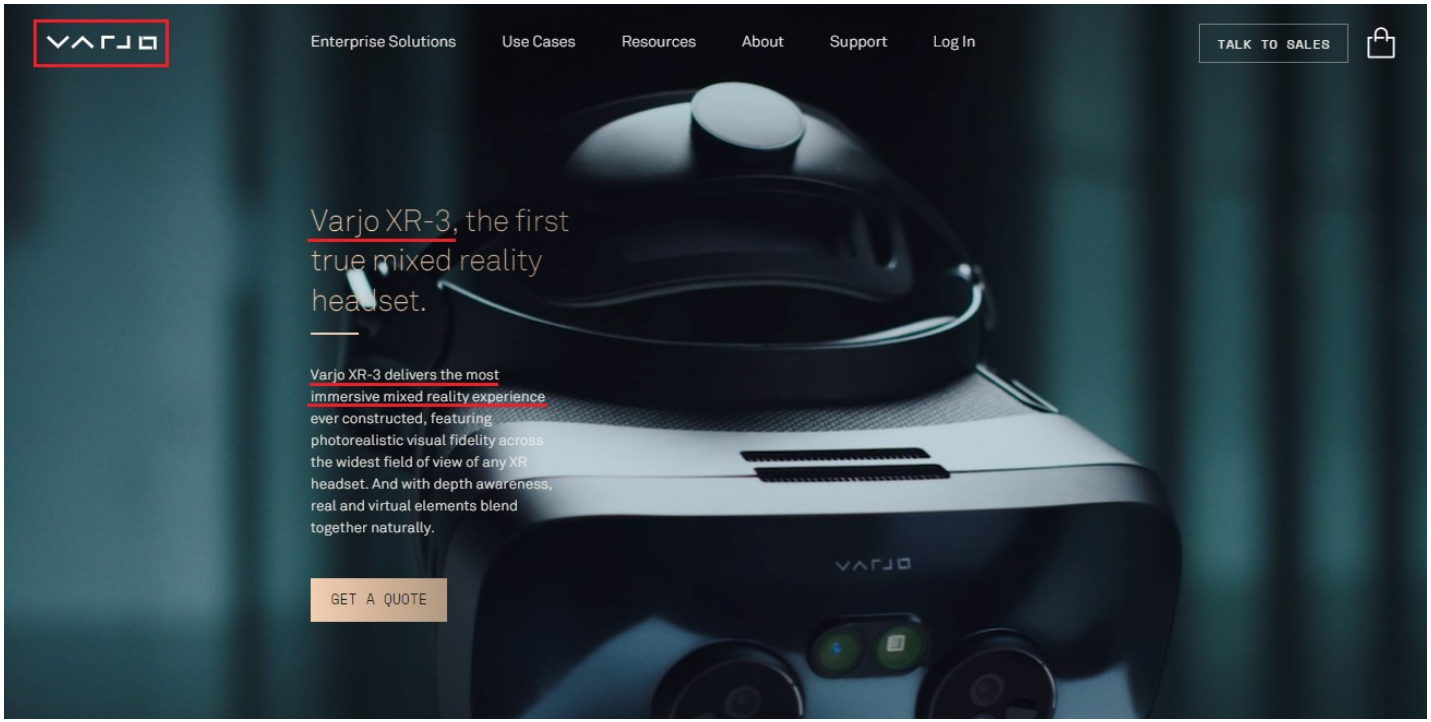
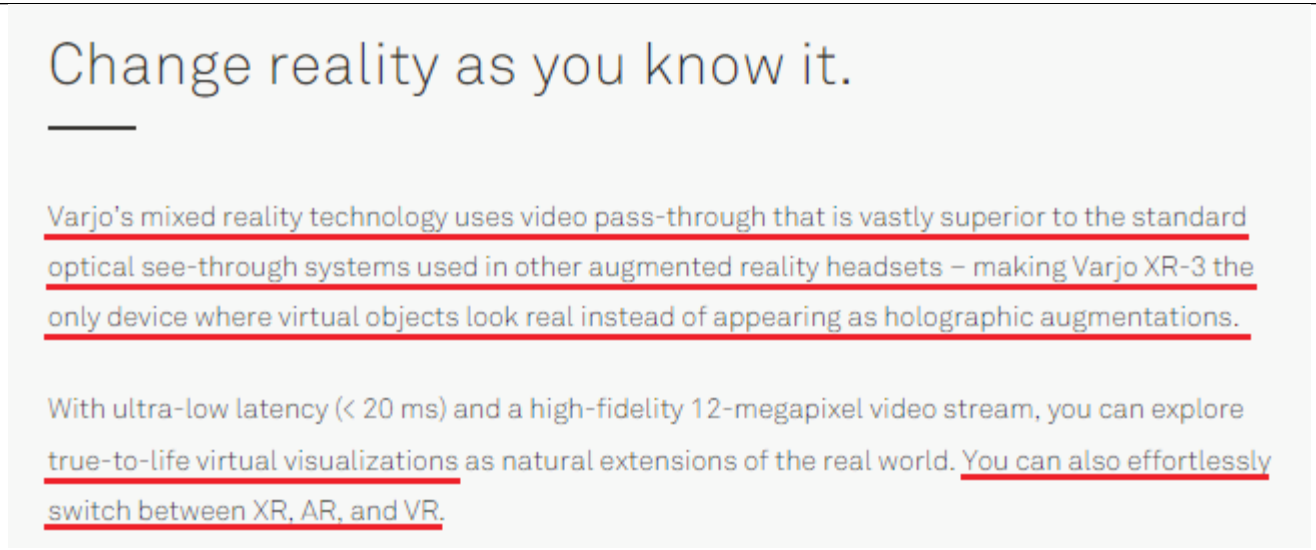
Exhibit 2

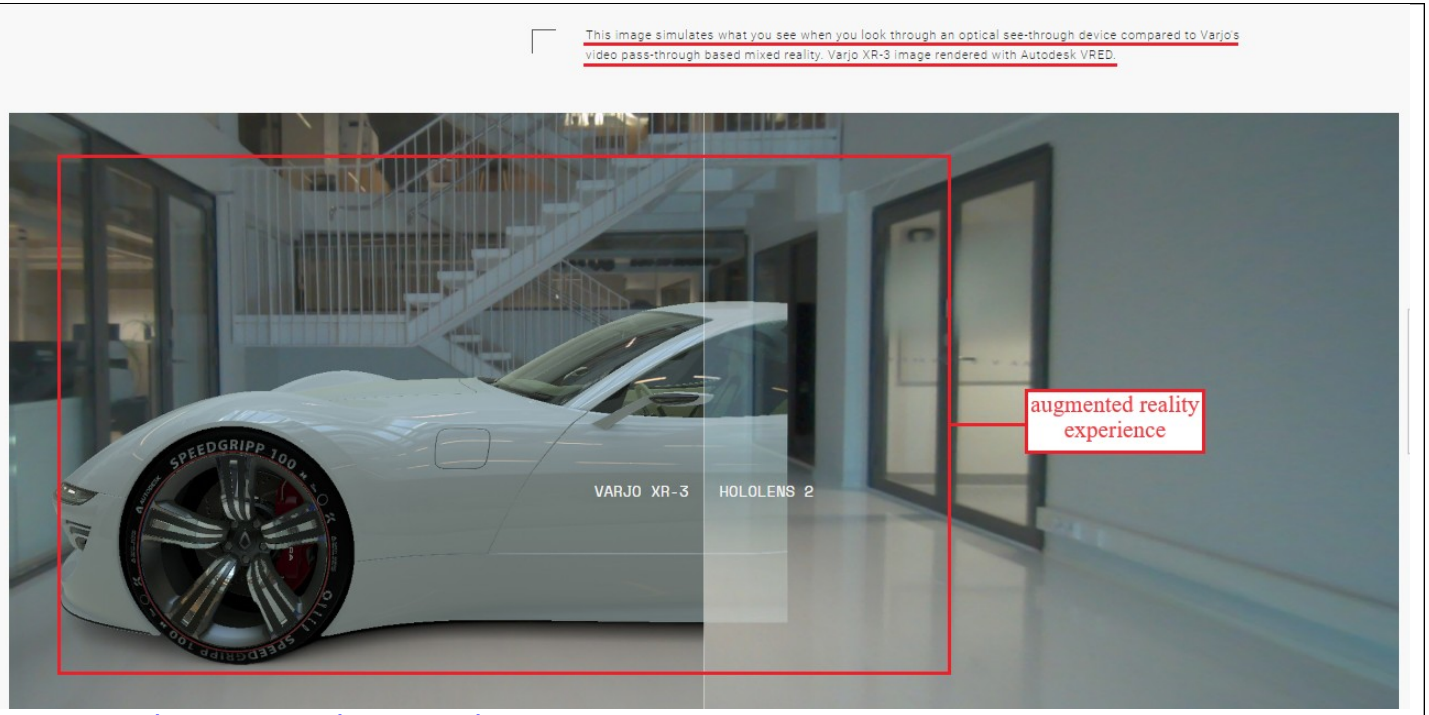
Method Claim: 1

US11080885	Varjo XR-3 (“The accused product”)
1. A method of providing an augmented reality experience with a user device, comprising:	The accused product discloses a method of providing an augmented reality experience with a user device (e.g., an XR headset such as Varjo XR-3). As shown below, a Varjo XR-3 headset provides an augmented reality experience to the user by overlaying virtual elements on top of the real-world view.  https://varjo.com/products/varjo-xr-3/

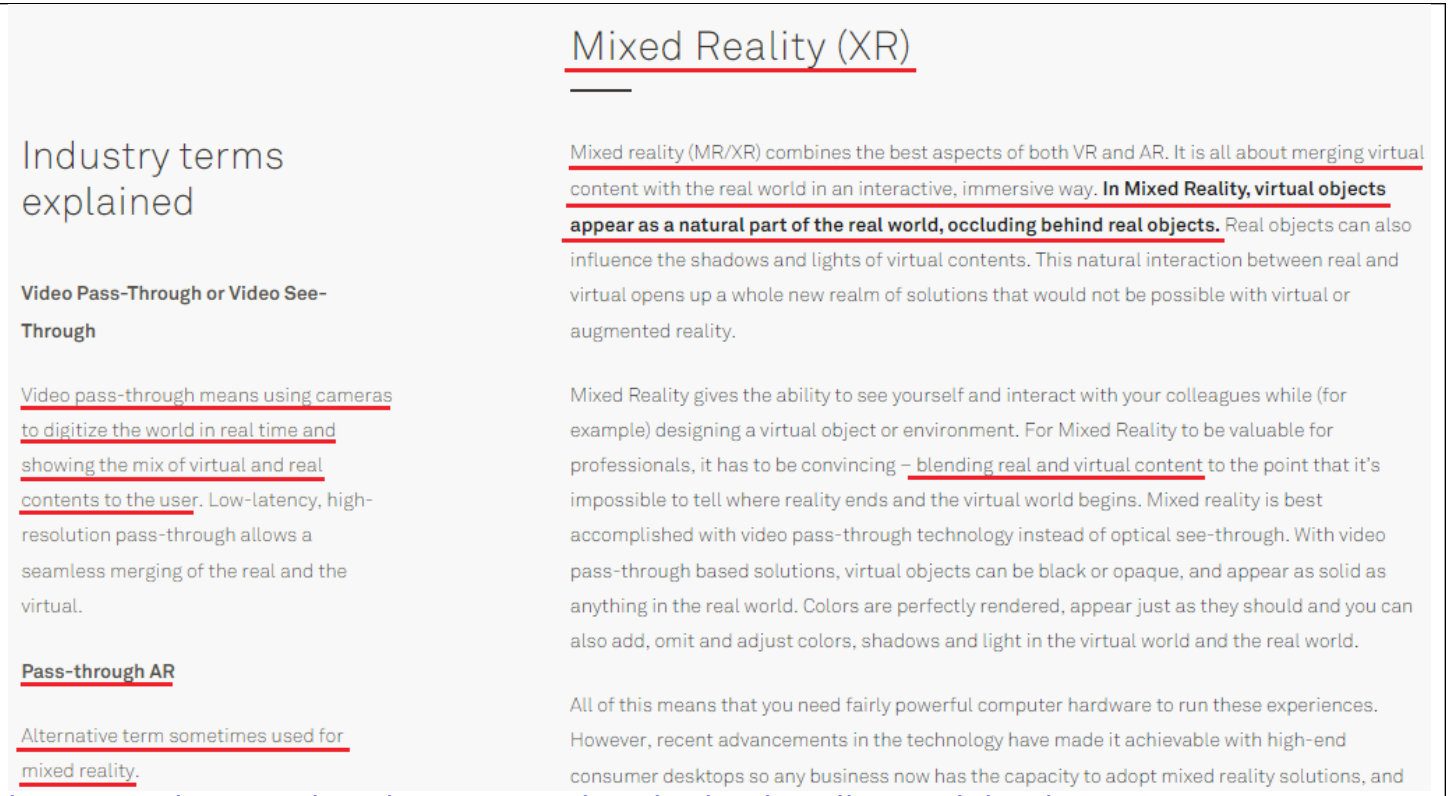


https://varjo.com/products/varjo-xr-3/

https://varjo.com/products/varjo-xr-3/



https://varjo.com/products/varjo-xr-3/



Industry terms explained

Video Pass-Through or Video See-Through

Video pass-through means using cameras to digitize the world in real time and showing the mix of virtual and real contents to the user. Low-latency, high-resolution pass-through allows a seamless merging of the real and the virtual.

Pass-through AR

Alternative term sometimes used for mixed reality.

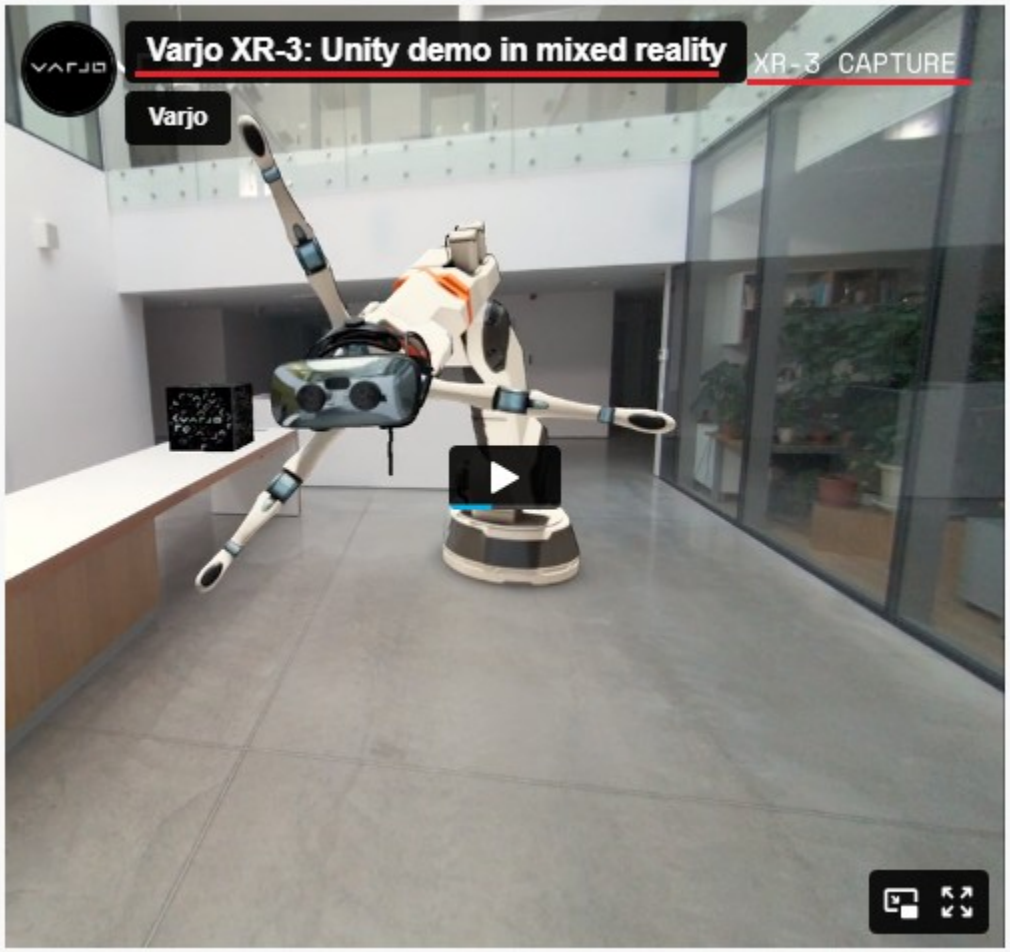
Mixed Reality (XR)

Mixed reality (MR/XR) combines the best aspects of both VR and AR. It is all about merging virtual content with the real world in an interactive, immersive way. **In Mixed Reality, virtual objects appear as a natural part of the real world, occluding behind real objects.** Real objects can also influence the shadows and lights of virtual contents. This natural interaction between real and virtual opens up a whole new realm of solutions that would not be possible with virtual or augmented reality.

Mixed Reality gives the ability to see yourself and interact with your colleagues while (for example) designing a virtual object or environment. For Mixed Reality to be valuable for professionals, it has to be convincing – blending real and virtual content to the point that it's impossible to tell where reality ends and the virtual world begins. Mixed reality is best accomplished with video pass-through technology instead of optical see-through. With video pass-through based solutions, virtual objects can be black or opaque, and appear as solid as anything in the real world. Colors are perfectly rendered, appear just as they should and you can also add, omit and adjust colors, shadows and light in the virtual world and the real world.

All of this means that you need fairly powerful computer hardware to run these experiences. However, recent advancements in the technology have made it achievable with high-end consumer desktops so any business now has the capacity to adopt mixed reality solutions, and

https://varjo.com/virtual-augmented-and-mixed-reality-explained/

	 https://varjo.com/virtual-augmented-and-mixed-reality-explained/
receiving an input image of a physical environment from a camera	The accused product discloses receiving an input image of a physical environment(e.g., an input image of the physical environment captured by the pass through camera of the Varjo XR3 headset), from a camera (e.g., a pass through camera of the Varjo XR3 headset) included in or on the user device (e.g., an XR headset such as Varjo XR-3), wherein a digitally encoded marker (DEM) (e.g., a Varjo Marker) is positioned at a

included in or on the user device, wherein a digitally encoded marker (DEM) is positioned at a marker location within the physical environment;	marker location within the physical environment (e.g., a location of the Varjo Marker placed within the environment). As shown below, a video is captured by the pass-through camera of the Varjo XR-3 headset. The physically printed Varjo Marker is positioned within the physical environment to track a physical object or surface. The position of the Varjo marker is relative to the tracking space of the Varjo XR-3.  https://varjo.com/products/varjo-xr-3/

Technical Specifications of Varjo XR-3

DISPLAY AND RESOLUTION	Full Frame Bionic Display with human-eye resolution. Focus area (27° x 27°) at 70 PPD uOLED, 1920 x 1920 px per eye Peripheral area at over 30 PPD LCD, 2880 x 2720 px per eye Colors: 99% sRGB, 93% DCI-P3
FIELD OF VIEW	Horizontal 115°
REFRESH RATE	90 Hz
MIXED REALITY	Ultra-low latency, dual 12-megapixel video pass-through at 90 Hz
POSITIONAL TRACKING	SteamVR™ 2.0 (recommended) or 1.0 tracking system Varjo inside-out tracking (beta) utilizing RGB video pass-through cameras

https://varjo.com/products/varjo-xr-3/

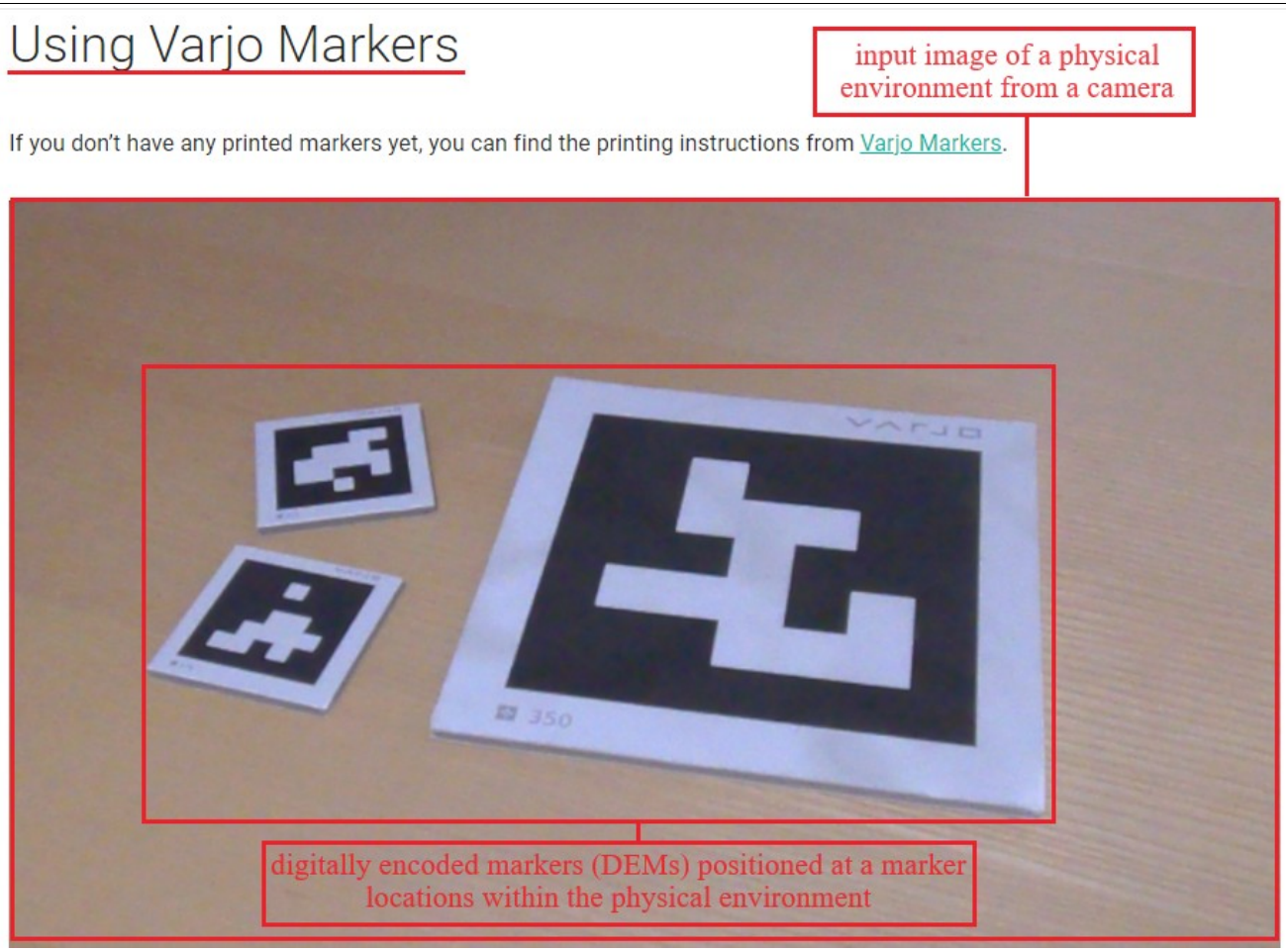
Varjo Markers

Varjo Markers are physical markers tracked by the video pass-through cameras on Varjo XR-3.

Different types of markers can be used for different purposes. As an example, Varjo Markers can be used as cheap replacements for electronic trackers such as Vive Trackers. The cost per printed tracker is significantly lower and the markers require no power to function.

https://developer.varjo.com/docs/get-started/varjo-markers

	# Varjo Markers Varjo Markers can be used to track static or dynamic real-world objects using the video pass-through cameras on Varjo XR-3. See Varjo Markers for more general information and printing instructions. Varjo Markers are typically used in mixed reality applications, but can be also used without video pass-through rendering in virtual reality applications to align virtual objects in the scene with physical objects in the real world. https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5



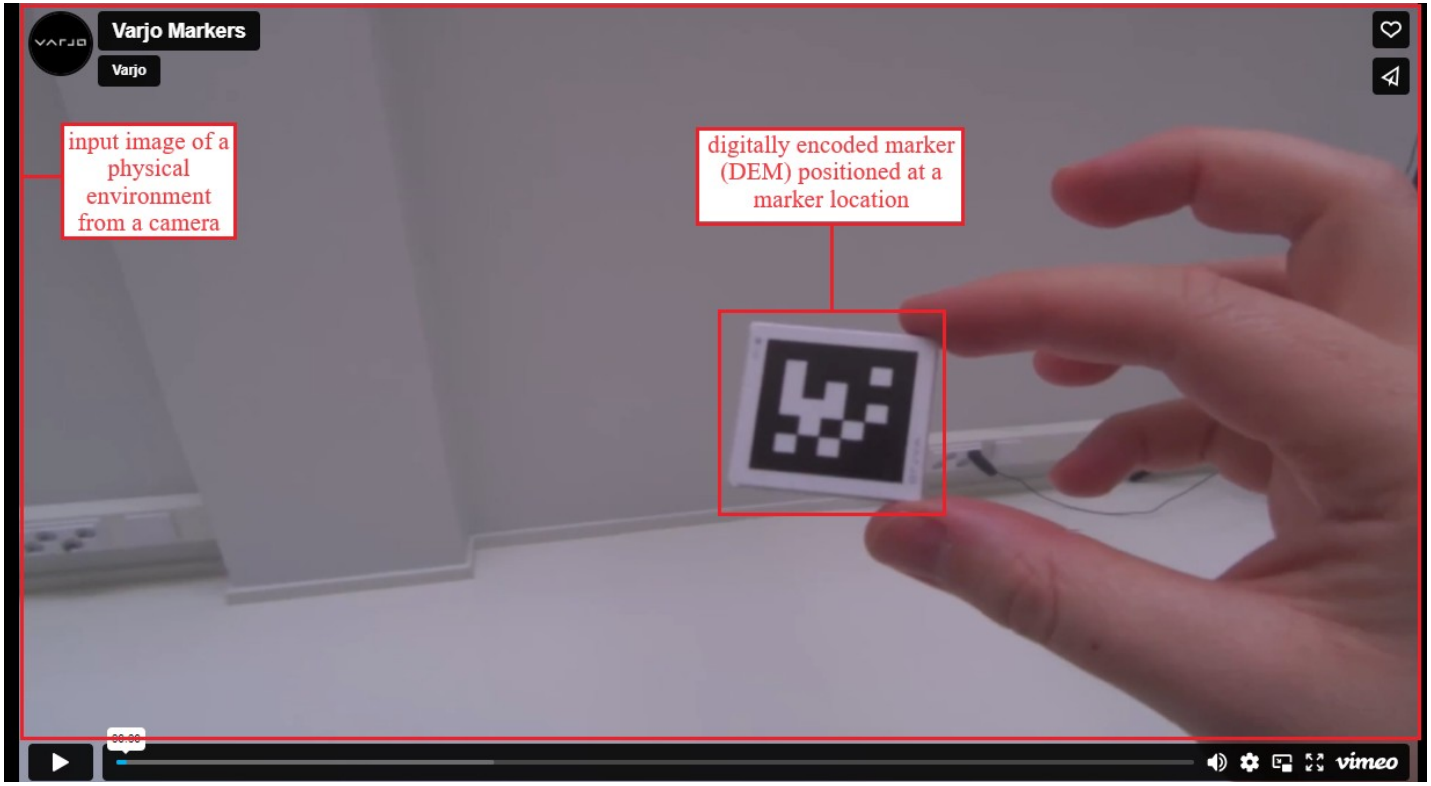
https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

Compile and **Save**. Once Varjo Marker tracking is enabled, you can retrieve updates with events. **New Varjo Marker Detected** is triggered when a new marker comes visible to the video pass-through cameras. **Varjo Marker Moved** triggers every time an already known marker receives an updated pose. **Varjo Marker Lost** is triggered when a known marker has been out of sight for a given timeout.

receiving input image from a camera

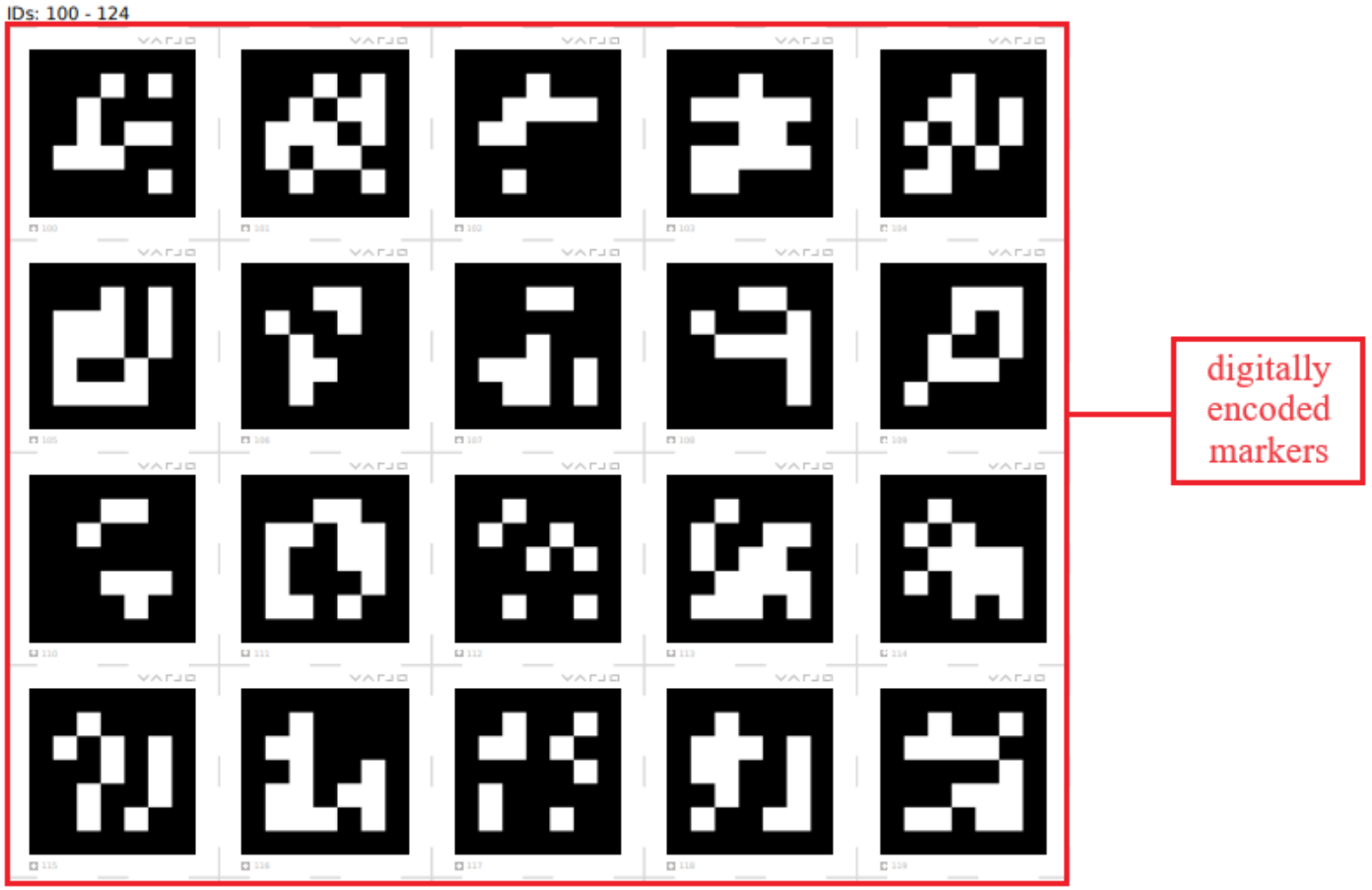
All events provide a *Marker ID*, which matches the ID on the printed marker. In addition, **New Varjo Marker Detected** and **Varjo Marker Lost** events provide *Position*, *Rotation* and *Size* for the marker. The size of a marker never changes and it's provided in Unreal units, taking *World to Meters scale* into account so the size always matches the physical marker.

https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5



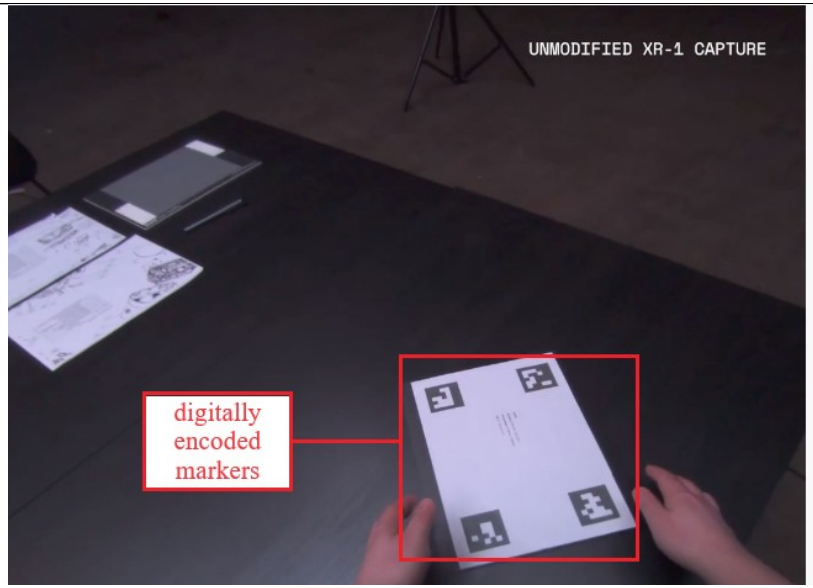
https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-

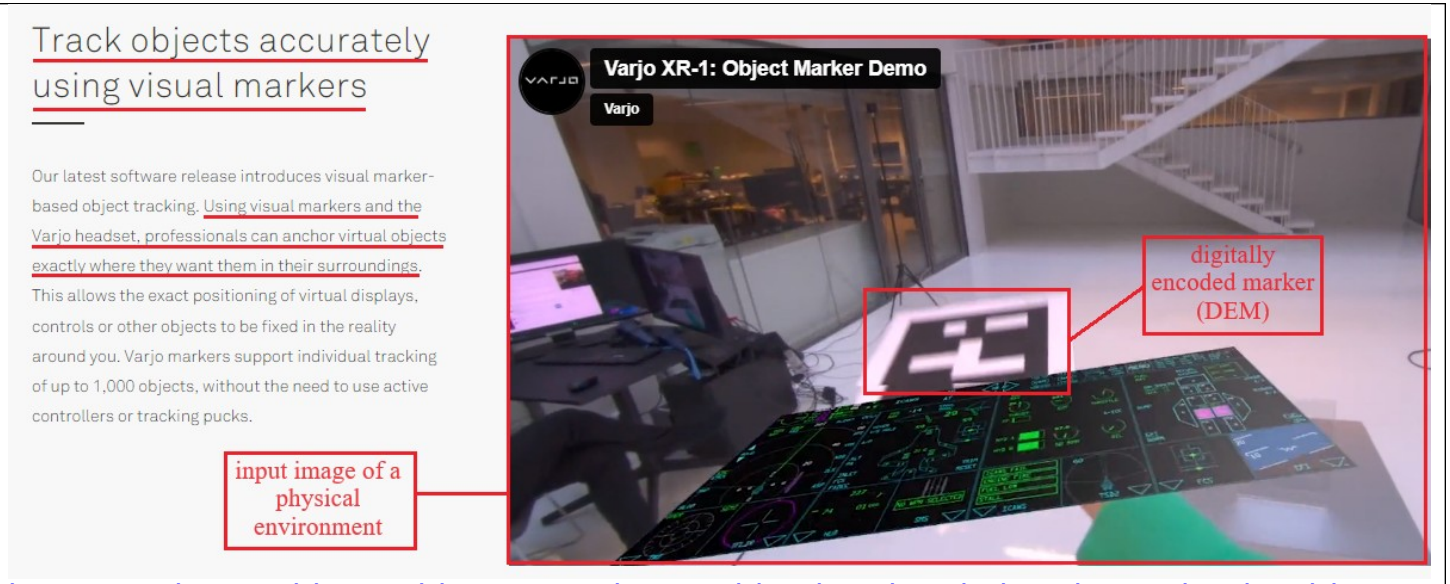
markers/



https://varjo-storage.s3.eu-central-1.amazonaws.com/docs/varjo-markers/VarjoMarkers-Object25mm-A4.pdf

	 https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

	 Varjo Markers come in three different sizes and the tracking distance varies upon the size of the marker: • **25mm marker, with an active area of up to 1 meter (3 feet).** The best usability is within 0.5 meters. These small markers are excellent for tracking something in your hand. • **50mm marker, with an active area of up to 2 meters (6.5 feet).** The best usability is within 1 meters. These are the most stable options for interactions within an arm's reach. • **150mm markers, with an active area of up to 5 meters (16 feet).** The best usability is within 2 meters for dynamic movement. Beyond 3 meters, we recommend setting a static position or using multiple markers to stabilize. In our 2.3 software update, we have implemented a predictive tracking mode for fast-moving objects. Users will be able to choose between standard and predictive tracking modes, enabling the most effective and accurate overall positioning based on the use case. https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

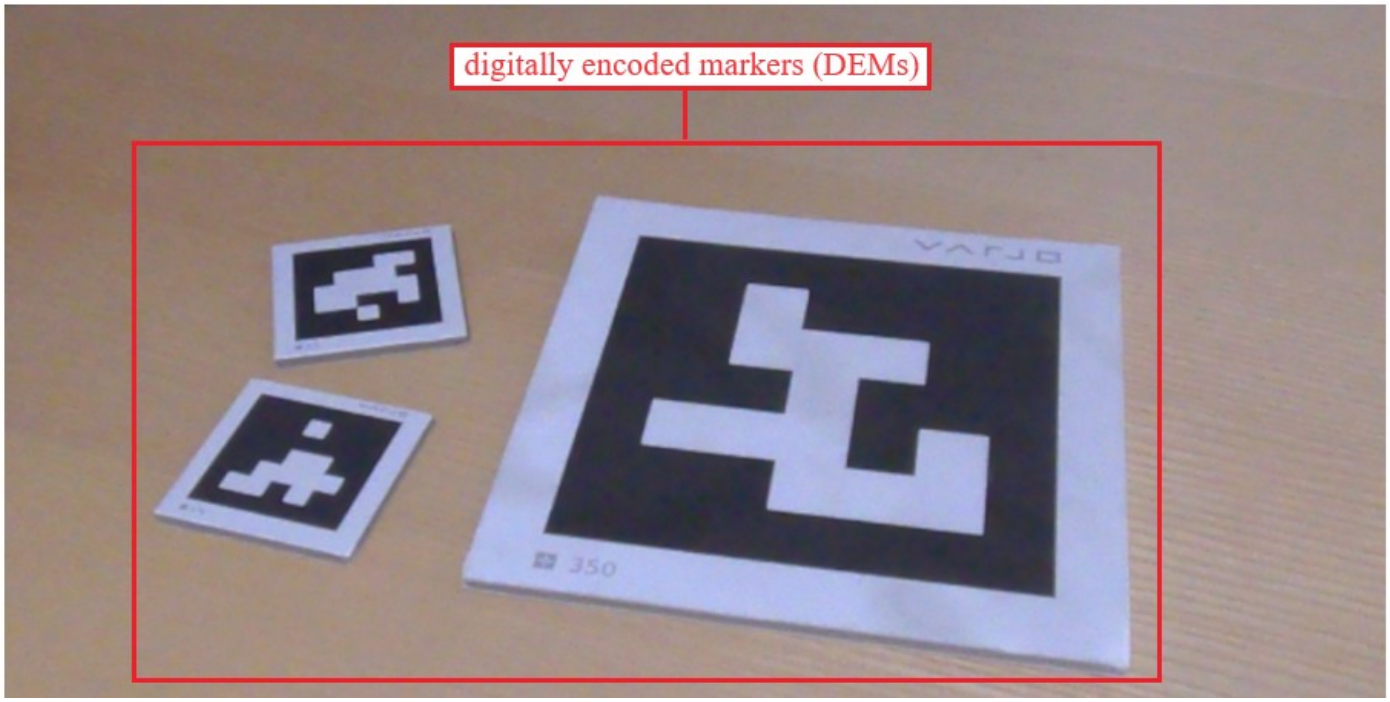
	 https://varjo.com/blog/webinar-recap-how-to-blend-real-and-virtual-seamlessly-with-new-xr-1-features/
decoding data from the DEM by processing the input image, wherein the decoded data comprises at least one of geographic coordinate data and relative coordinate data;	The accused product discloses decoding data (e.g., pose data for a Varjo Marker) from the DEM (e.g., a Varjo Marker) by processing the input image (e.g., an input image of the physical environment captured by the pass through camera of the Varjo XR3 headset), wherein the decoded data (e.g., pose data for a Varjo Marker) comprises at least one of geographic coordinate data and relative coordinate data (e.g., marker pose data). As shown below, the Varjo XR-3 headset tracks the Varjo Marker in the pass-through camera video feed. In the Varjo Native SDK, the position of the center of the marker is obtained via varjo_WorldObjectMarkerComponent, varjo_PoseComponent and varjo_WorldPoseComponent. The varjo_WorldObjectMarkerComponent represents a marker identified in the camera stream. varjo_WorldGetPoseComponent and varjo_WorldGetVisualMarkerComponent are used to get the marker pose and other marker-specific data.

	Varjo SDK mentions that the pose of a Varjo Marker is relative to the tracking space. Further, for Unreal Engine, a new Varjo Marker along with its Position is detected via the New Varjo Marker Detected of the VarjoMarkersEvent. If the Varjo Marker has moved, the new position is detected via the Varjo Marker Moved of the VarjoMarkersEvent. The position obtained from both the events is the relative coordinate of the marker in the tracking space. Varjo Markers Varjo Markers are physical markers tracked by the video pass-through cameras on Varjo XR-3. Different types of markers can be used for different purposes. As an example, Varjo Markers can be used as cheap replacements for electronic trackers such as Vive Trackers. The cost per printed tracker is significantly lower and the markers require no power to function. https://developer.varjo.com/docs/get-started/varjo-markers

	# Varjo Markers Varjo Markers can be used to track static or dynamic real-world objects using the video pass-through cameras on Varjo XR-3. See Varjo Markers for more general information and printing instructions. Varjo Markers are typically used in mixed reality applications, but can be also used without video pass-through rendering in virtual reality applications to align virtual objects in the scene with physical objects in the real world. https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

Using Varjo Markers

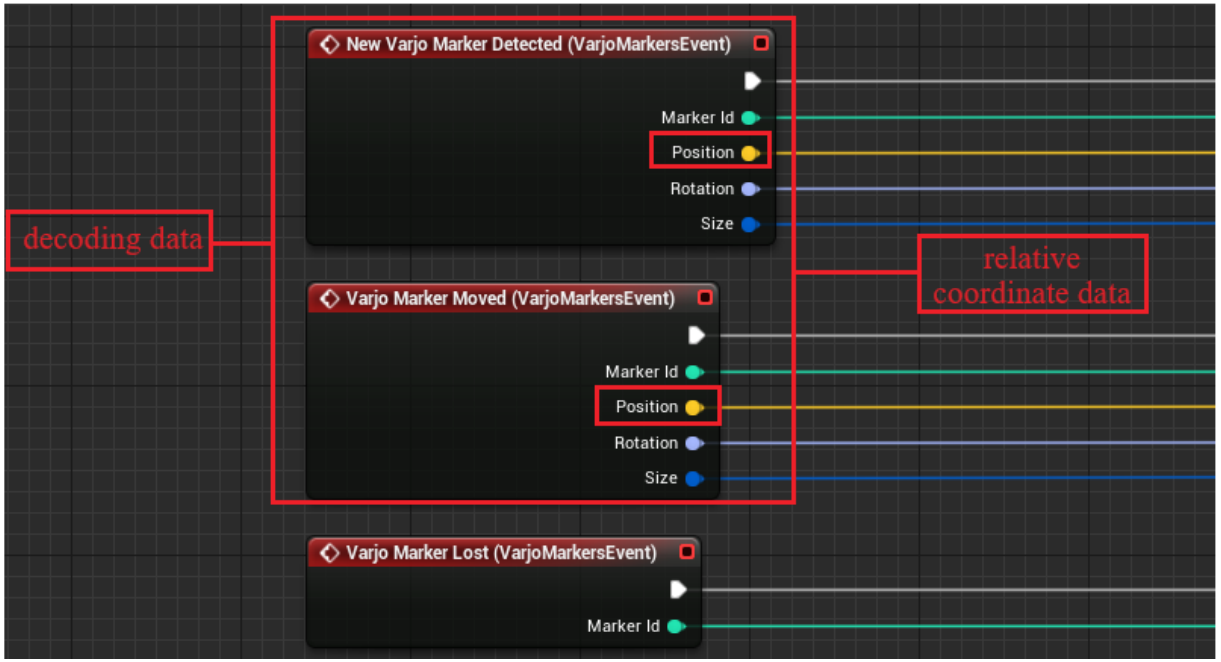
If you don't have any printed markers yet, you can find the printing instructions from Varjo Markers.



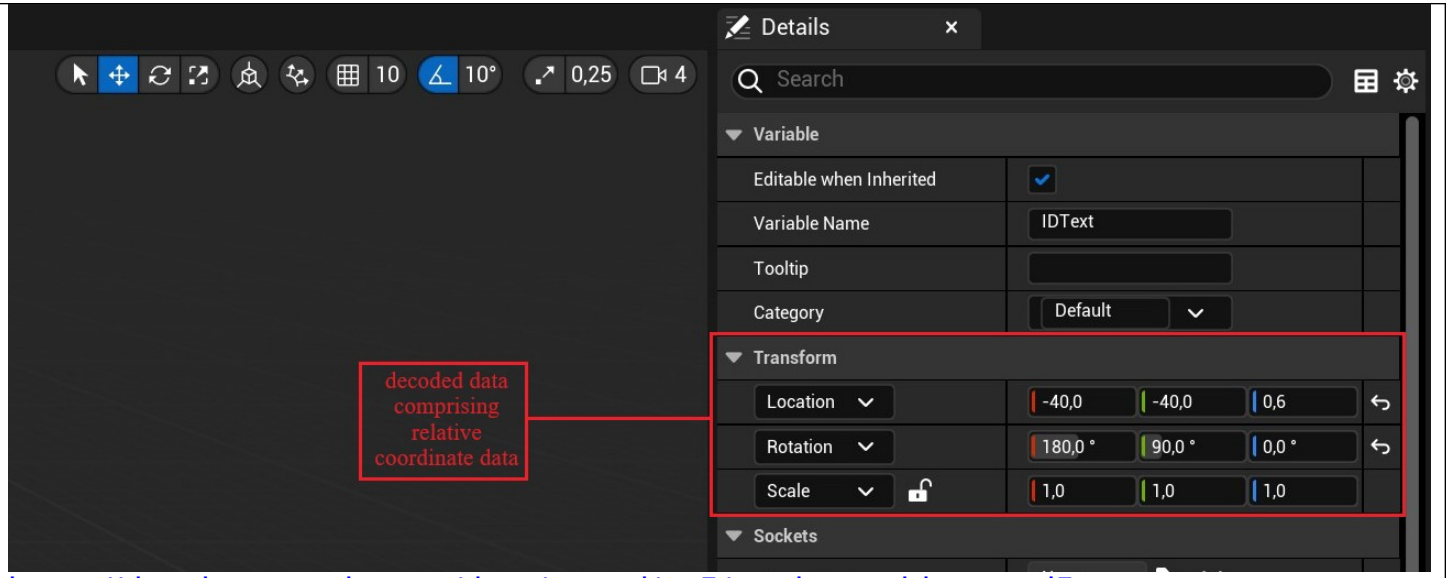
https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

Compile and **Save**. Once Varjo Marker tracking is enabled, you can retrieve updates with events. **New Varjo Marker Detected** is triggered when a new marker comes visible to the video pass-through cameras. **Varjo Marker Moved** triggers every time an already known marker receives an updated pose. **Varjo Marker Lost** is triggered when a known marker has been out of sight for a given timeout.

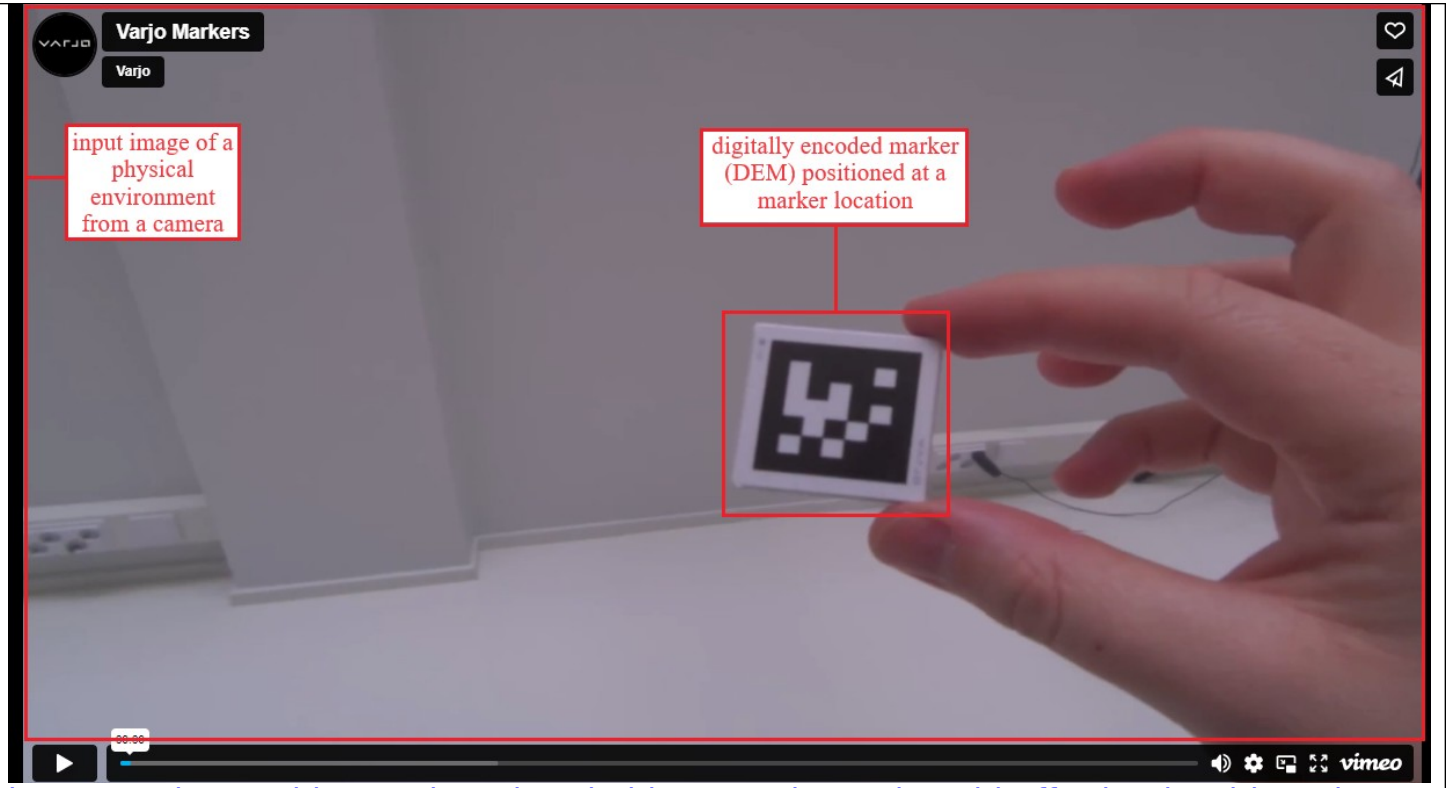
All events provide a *Marker ID*, which matches the ID on the printed marker. In addition, **New Varjo Marker Detected** and **Varjo Marker Lost** events provide *Position*, *Rotation* and *Size* for the marker. The size of a marker never changes and it's provided in Unreal units, taking *World to Meters scale* into account so the size always matches the physical marker.



https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

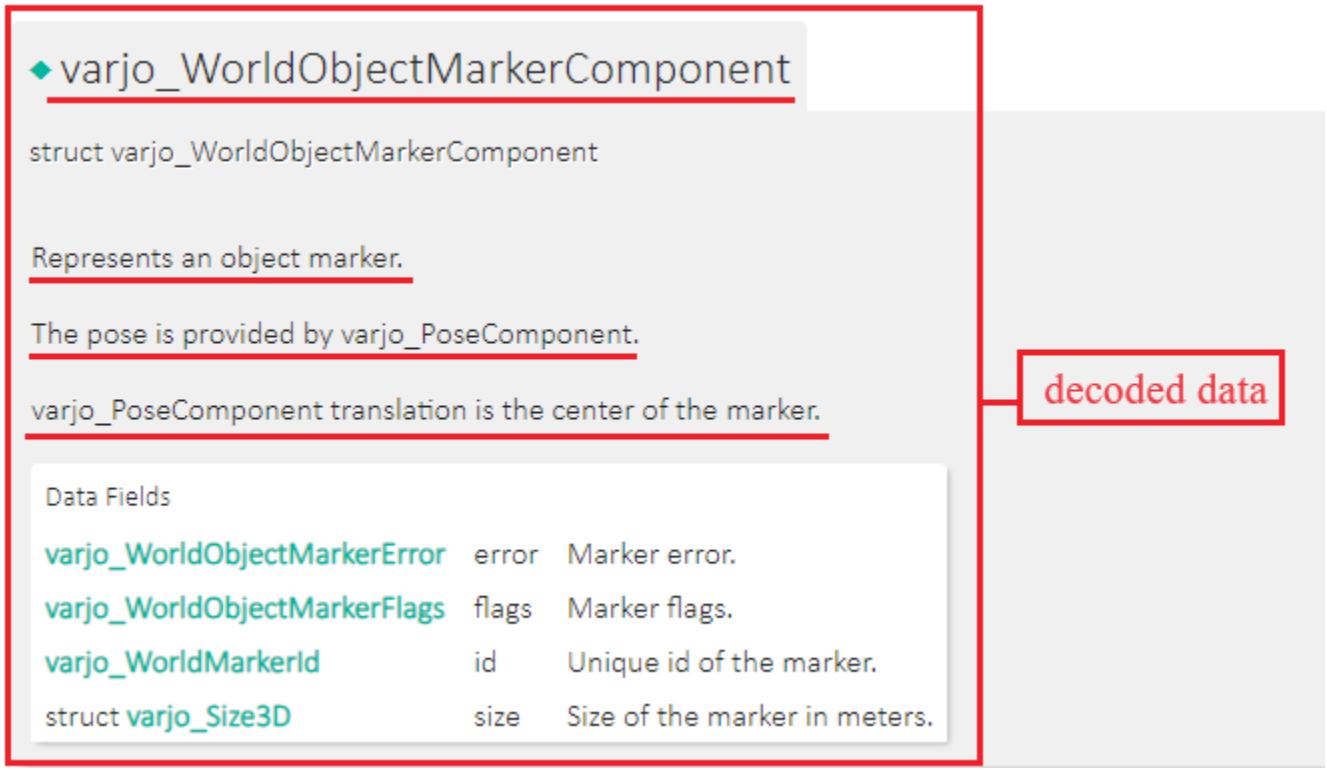


https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

	 https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

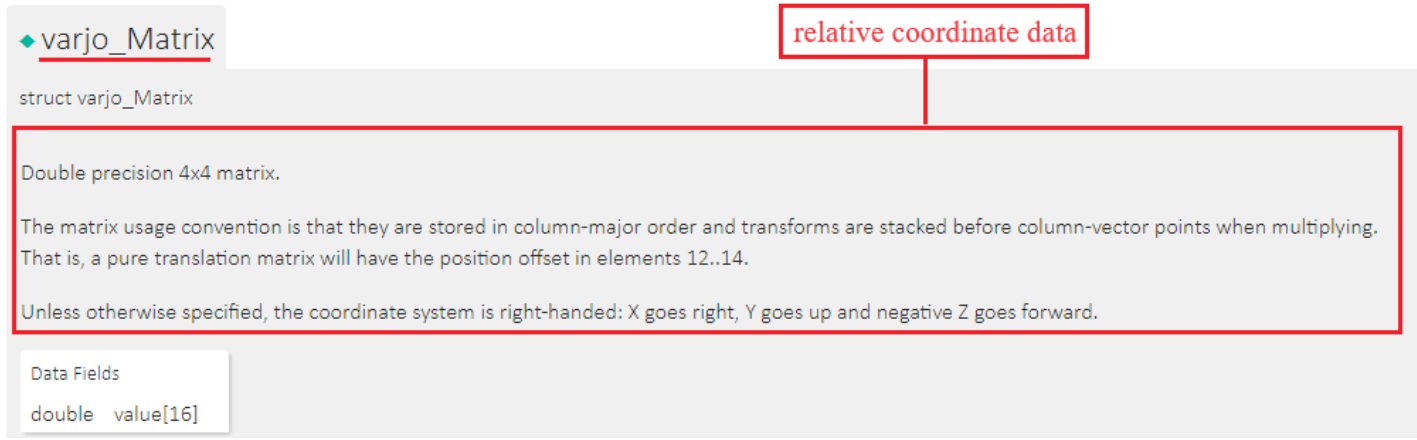
	 https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

	# Varjo Marker example An example of using Varjo marker can be found in the examples folder included in the Native SDK package. Open the MarkerExample folder and study the **MarkerScene.cpp** file to see the basic Varjo marker implementation. The example application enables video pass-through and allows the user to see a virtual marker in place of the Varjo marker in their view. In suitable conditions (with good lighting, no reflections, no bending or curing, and a correctly-sized printed marker), the Varjo marker and virtual marker should align perfectly. The example application also allows a user to preview how many markers are visible in the view at any given time. **varjo_WorldGetObjectCount** is used for the marker count. **varjo_WorldGetObjects** is used to get the marker objects. **varjo_WorldGetPoseComponent** and **varjo_WorldGetVisualMarkerComponent** are used to get the marker pose and other marker-specific data. https://developer.varjo.com/docs/native/varjo-sdk-examples#varjo-marker-example



https://developer.varjo.com/docs/v3.3.0/apidocs/_varjo__types__world_8h.html#structvarjo___world_object_marker_component

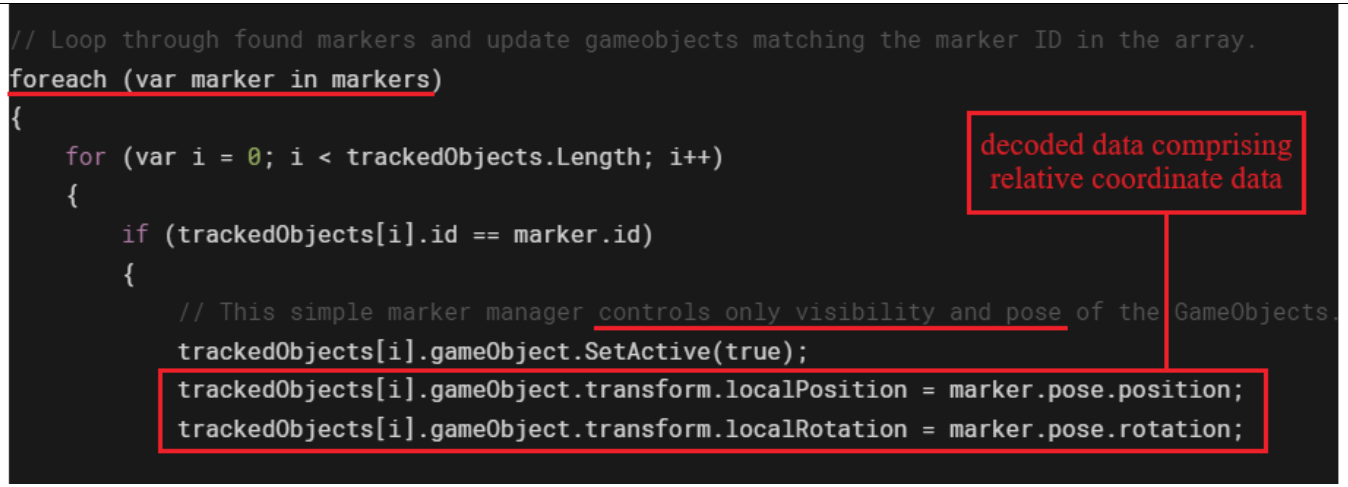
https://developer.varjo.com/docs/v3.3.0/apidocs/_varjo__types__world_8h.html#structvarjo___world_object_marker_component



https://developer.varjo.com/docs/v3.3.0/apidocs/_varjo__types_8h.html#structvarjo___matrix

Varjo Markers have two tracking modes; *Stationary* and *Dynamic*. The default tracking mode is *Stationary*, which means the pose is heavily filtered and it's less responsive to marker's movements. This is optimal, if you want to align fixed physical elements like a cockpit on a flight simulator with the virtual elements. When the tracking mode is *Dynamic*, the pose updates are more responsive to marker's movements. This should be used, if the marker is expected to move. Use *DoPrediction* flag to change the tracking mode. You can enable dynamic tracking with **VarjoMarkers.AddVarjoMarkerFlags(marker.id, VarjoMarkerFlags.DoPrediction)** and disable it with **VarjoMarkers.RemoveVarjoMarkerFlags(marker.id, VarjoMarkerFlags.DoPrediction)**.

https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin



https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin

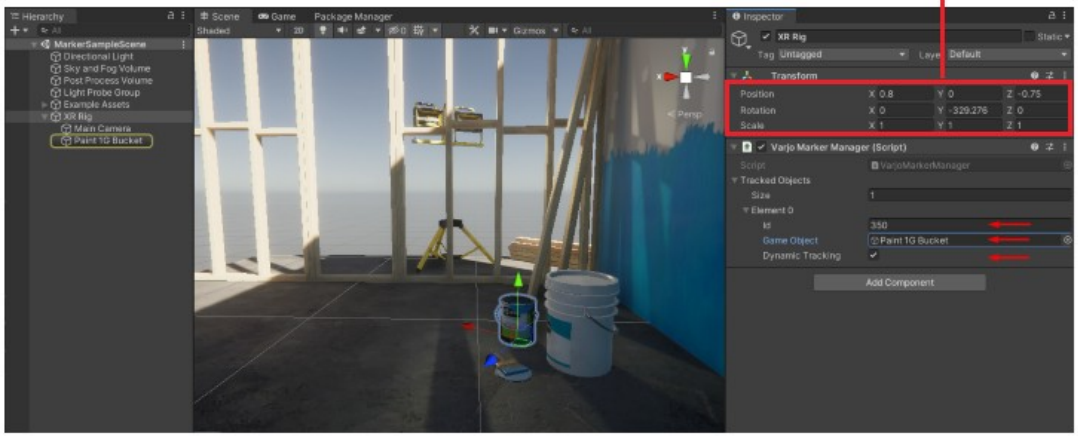
As the pose in the Varjo Marker is relative to the tracking space, you probably want to make the GameObject a sibling of the XR camera. Make sure that the GameObject is not flagged as static.

Add the GameObject in the array of TrackedObjects in your VarjoMarkerManager and set the ID matching the ID of a printed Varjo Marker.

If the marker is intended to be moved, enable Dynamic Tracking.

You can have as many tracked objects as you want as long you don't use multiple printed markers with the same ID in the environment.

decoded data comprises relative coordinate data



https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin

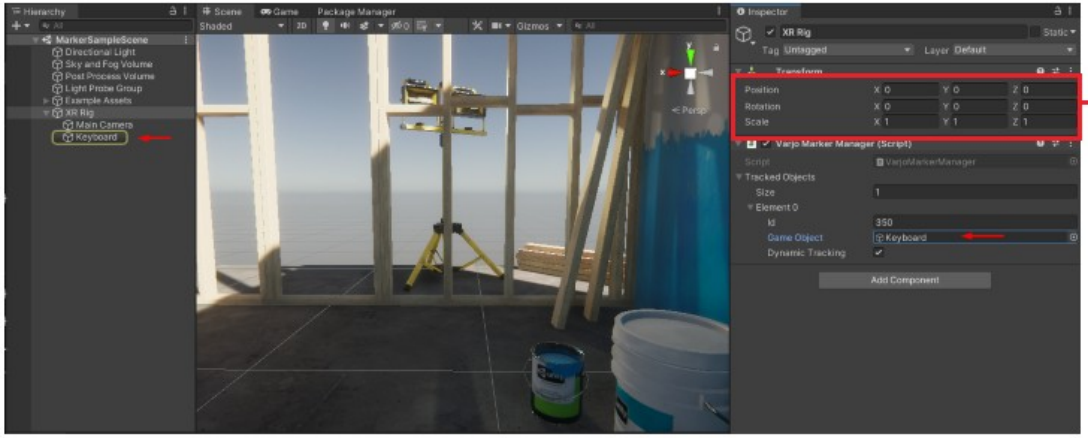
Using Varjo Markers to align a video pass-through mask with a real-world object

One of the most common uses for Varjo Markers is aligning a video pass-through mask with a physical object in the environment. This is useful when bringing objects such as input devices or instruments from the real world into the virtual environment.

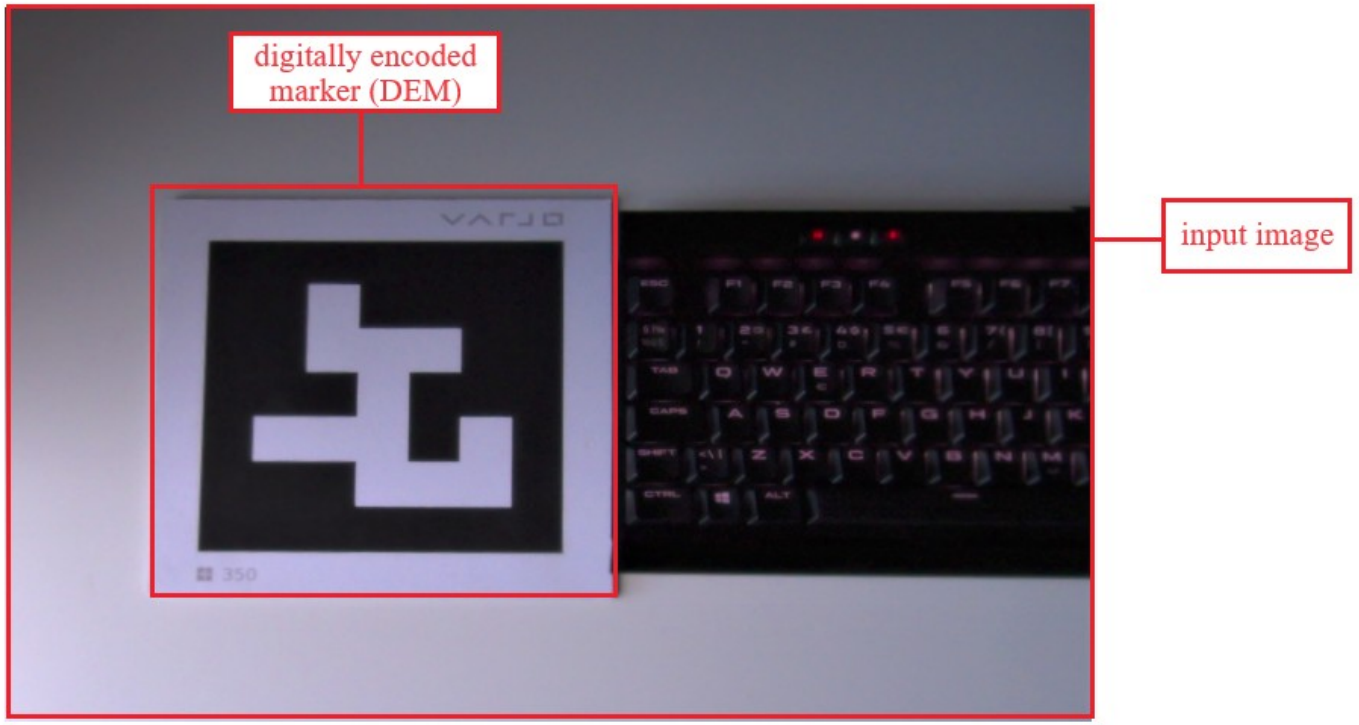
https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin

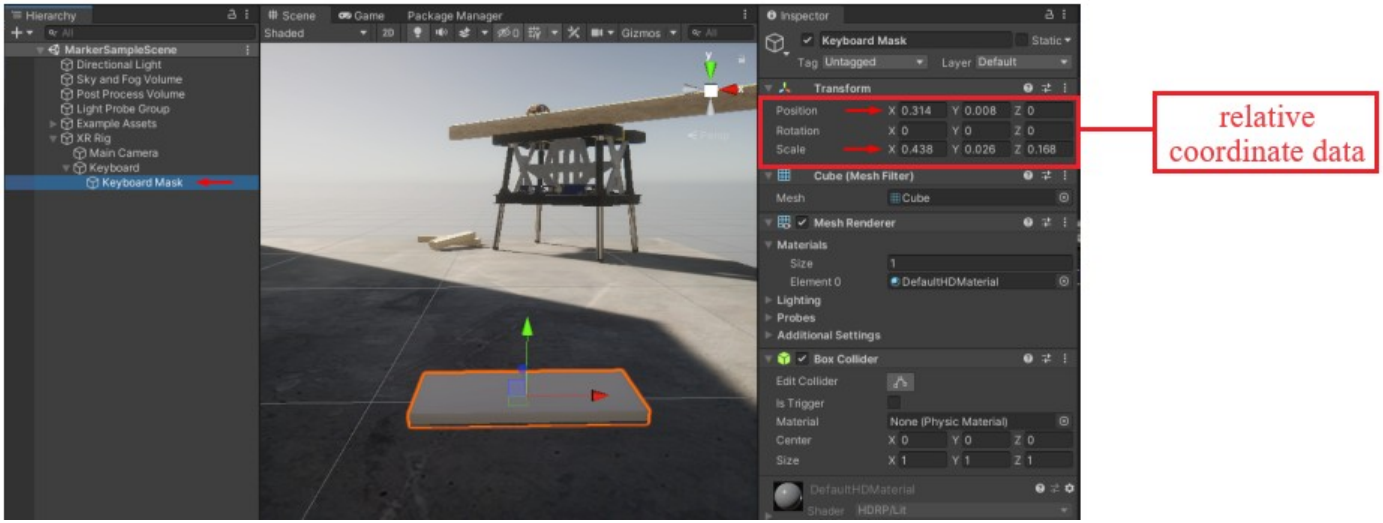
Continuing from the instructions above, replace a tracked GameObject with an empty GameObject. Add this GameObject in the TrackedObjects array with an ID matching your printed Varjo Marker.

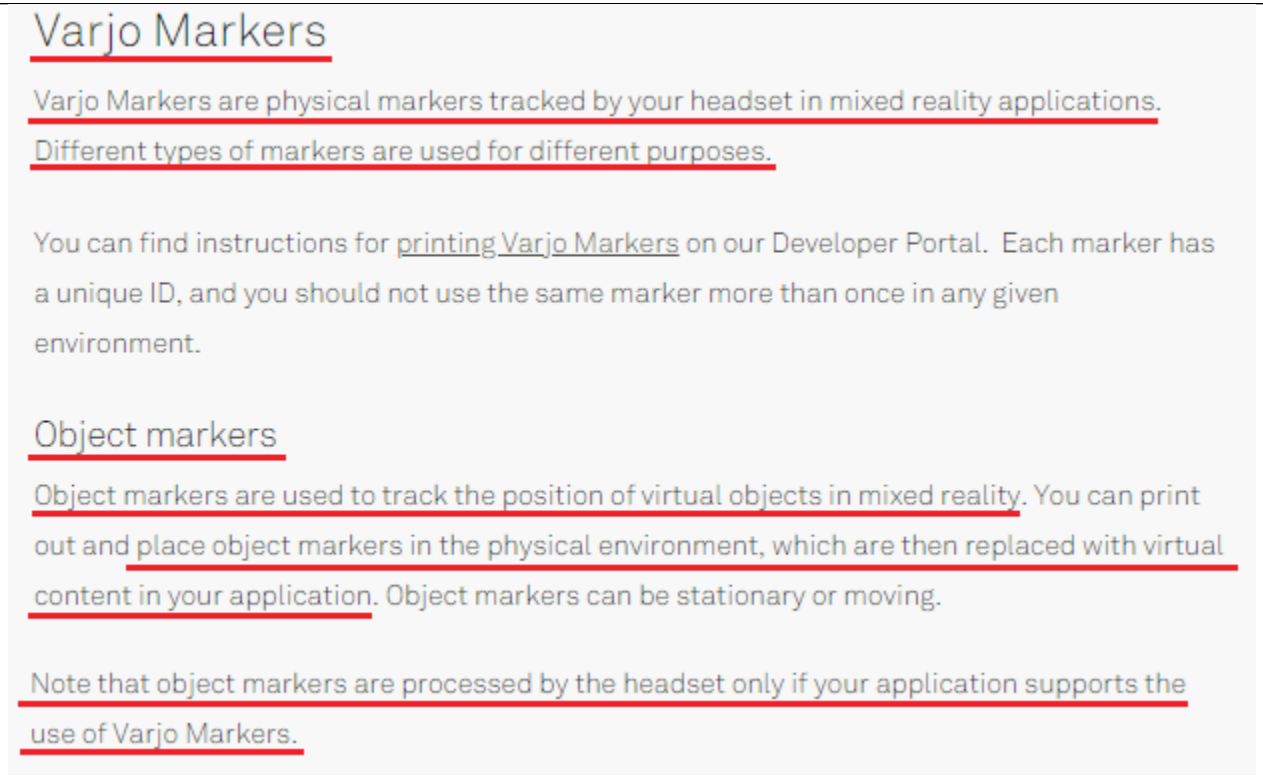
If the keyboard is intended to be moved, enable Dynamic Tracking. When masking stationary objects, it's better to keep Dynamic Tracking disabled.



https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin

	Place the printed marker next to your keyboard. Typically you would want to attach the marker to the physical object, so that they move together.  https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin

	In the Unity Editor, add a GameObject with a mesh for the mask as a child of the tracked GameObject. For the best quality you would want the mesh to be the same shapes as the physical object. For the sake of simplicity, we will use a Cube primitive and simply match the scale of the GameObject with the size of the keyboard. Measure the offset from the center of the printed Varjo Marker to the center of your keyboard. Set this offset as the local position of your mask.  https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin
retrieving digital content for a virtual object; and	The accused product discloses retrieving digital content (e.g., digital content such as a digital file of a virtual object) for a virtual object (e.g., a virtual object augmented over the Varjo Marker). As shown below, digital contents for a virtual object are retrieved and augmented over a Varjo Marker. Unreal Engine uses the Agent class to retrieve digital content for a virtual object to be augmented over a Varjo Marker.



Varjo Markers

Varjo Markers are physical markers tracked by your headset in mixed reality applications. Different types of markers are used for different purposes.

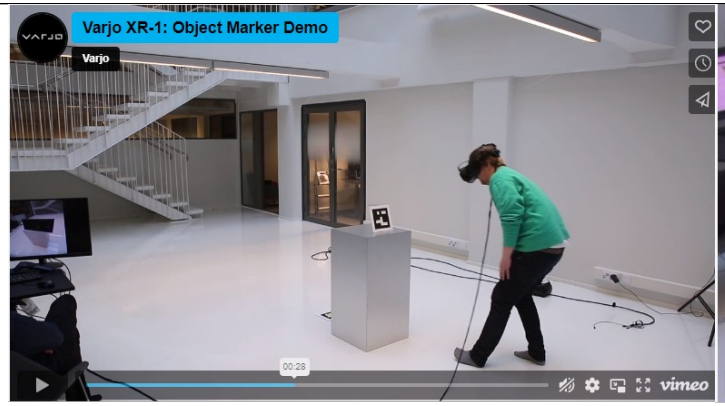
You can find instructions for printing Varjo Markers on our Developer Portal. Each marker has a unique ID, and you should not use the same marker more than once in any given environment.

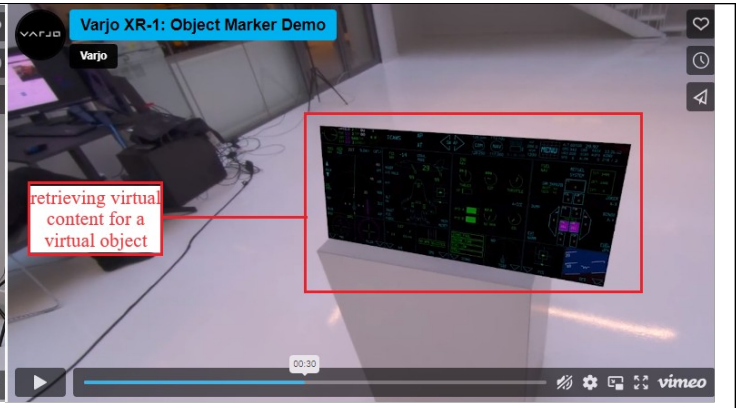
Object markers

Object markers are used to track the position of virtual objects in mixed reality. You can print out and place object markers in the physical environment, which are then replaced with virtual content in your application. Object markers can be stationary or moving.

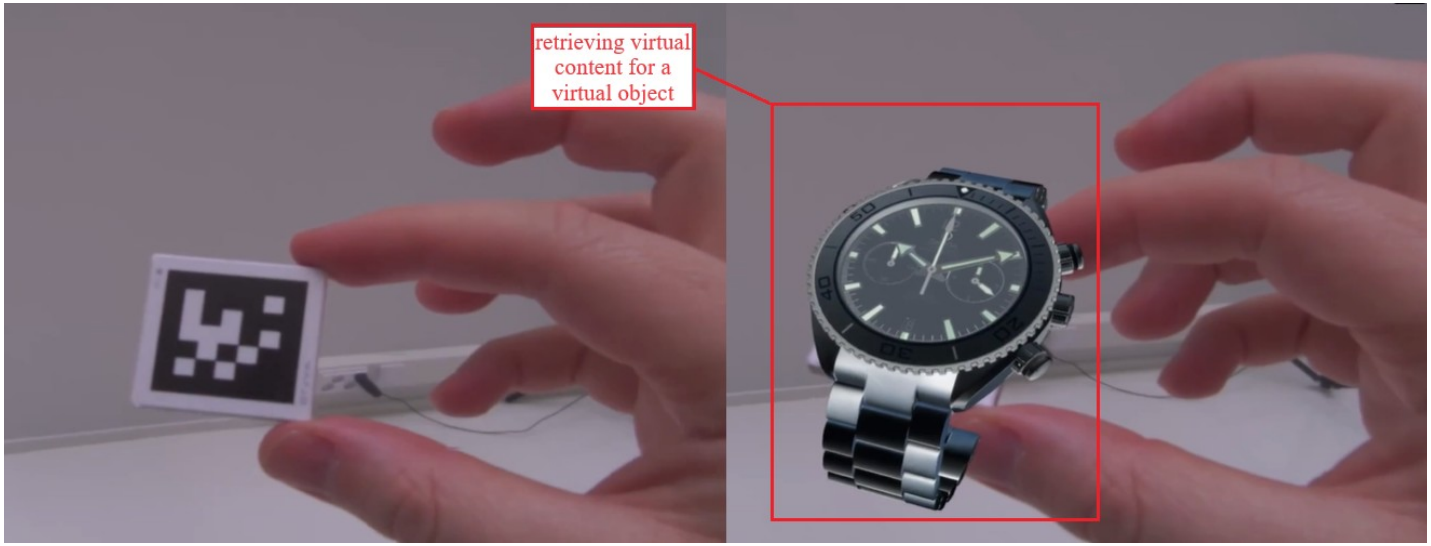
Note that object markers are processed by the headset only if your application supports the use of Varjo Markers.

https://varjo.com/use-center/get-to-know-your-headset/mixed-reality/#reference-marker





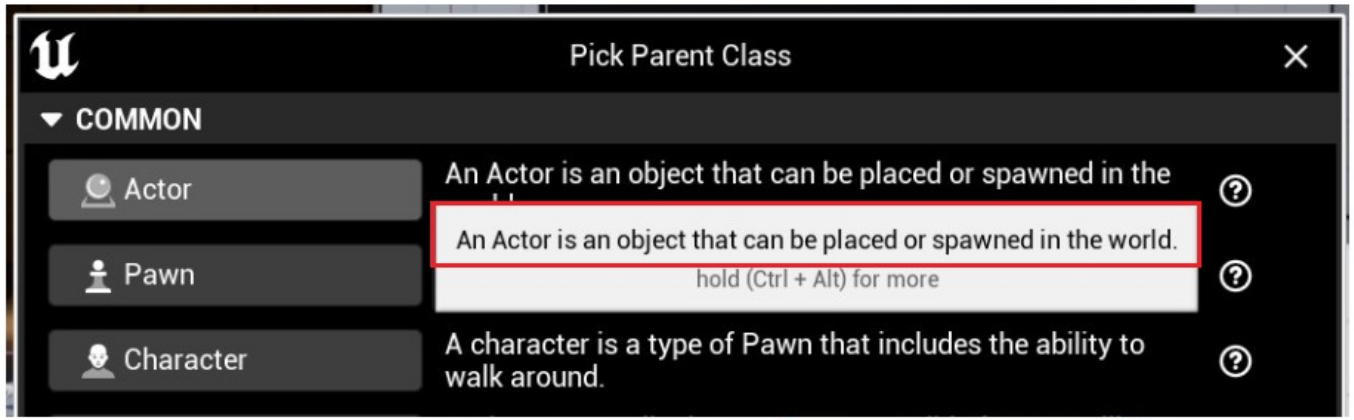
https://varjo.com/blog/webinar-recap-how-to-blend-real-and-virtual-seamlessly-with-new-xr-1-features/



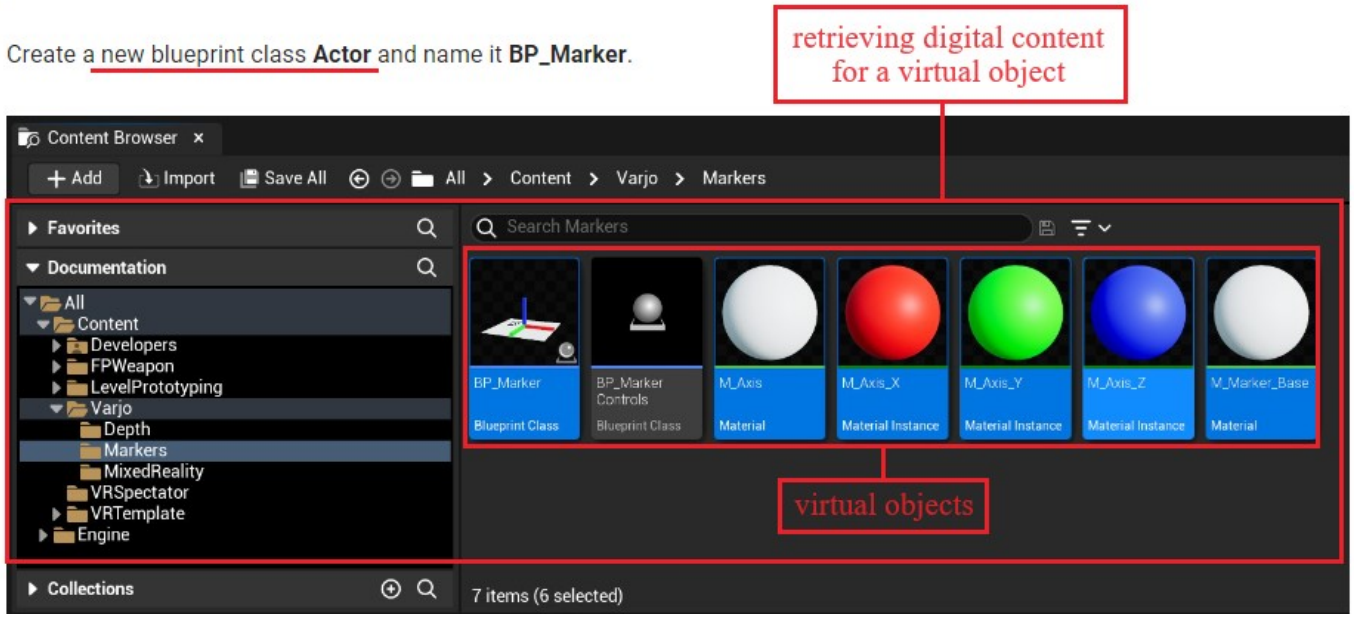
https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

Enable Varjo Marker tracking with **Set Varjo Marker Tracking Enabled**. Blueprint node **Is Varjo Marker Tracking Enabled** returns the current state.

Create a logic for this. Follow the instructions below. First, create a new blueprint as **Actor** and name it **BP_MarkerControls**.



Create a new blueprint class **Actor** and name it **BP_Marker**.



https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

Next you need to add a way to define GameObjects that will be following the tracked markers. For example, you can create a simple array of structs to store ID-GameObject pairs that can be edited in the Unity Inspector. We also include a boolean for controlling the tracking mode of the marker. Add this piece of code in the beginning of the *VarjoMarkerManager* class.

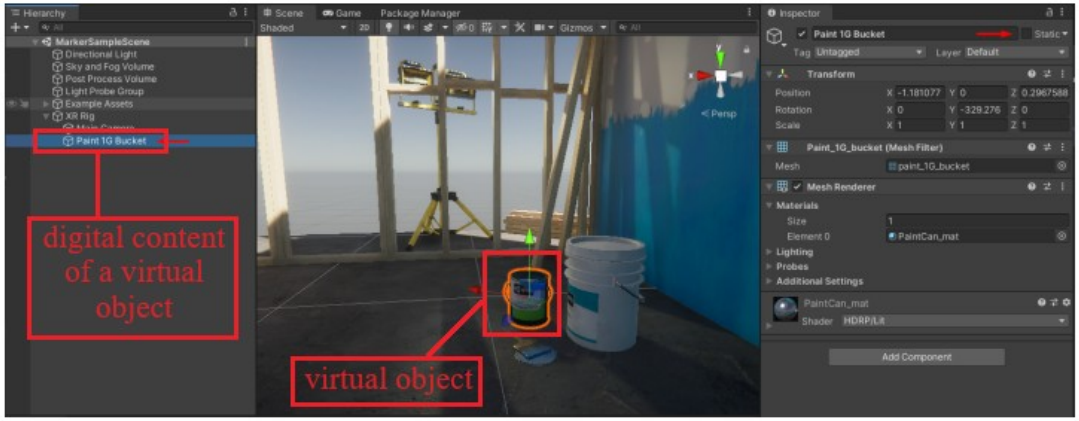
```
// Serializable struct to make it easy to add tracked objects in the Inspector.
[Serializable]
public struct TrackedObject
{
    public long id;
    public GameObject gameObject;
    public bool dynamicTracking;
}

// An public array for all the tracked objects.
public TrackedObject[] trackedObjects = new TrackedObject[1];
```

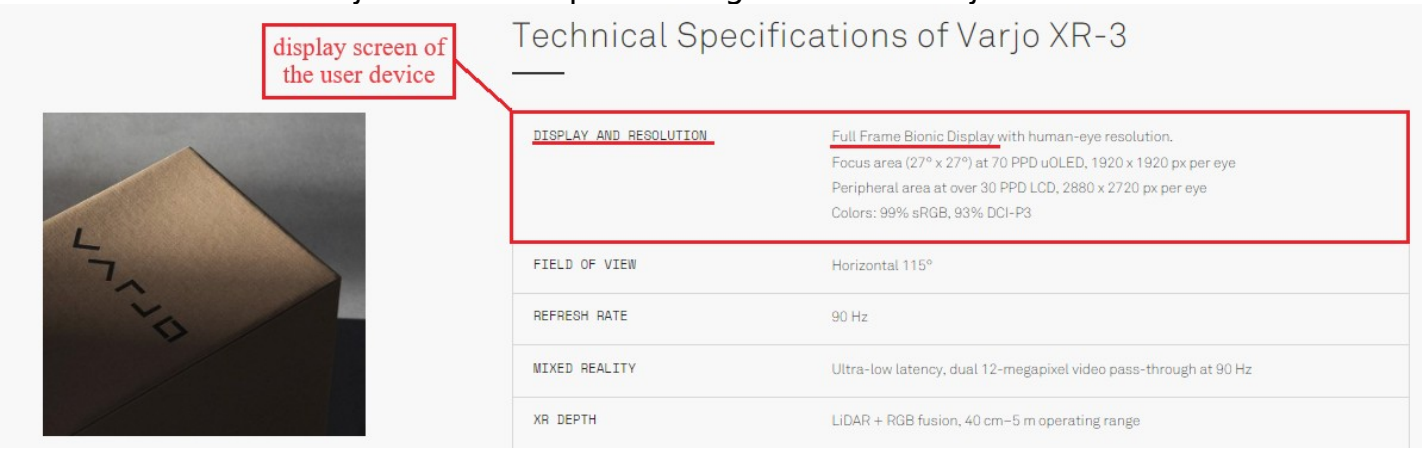
digital content for a virtual object

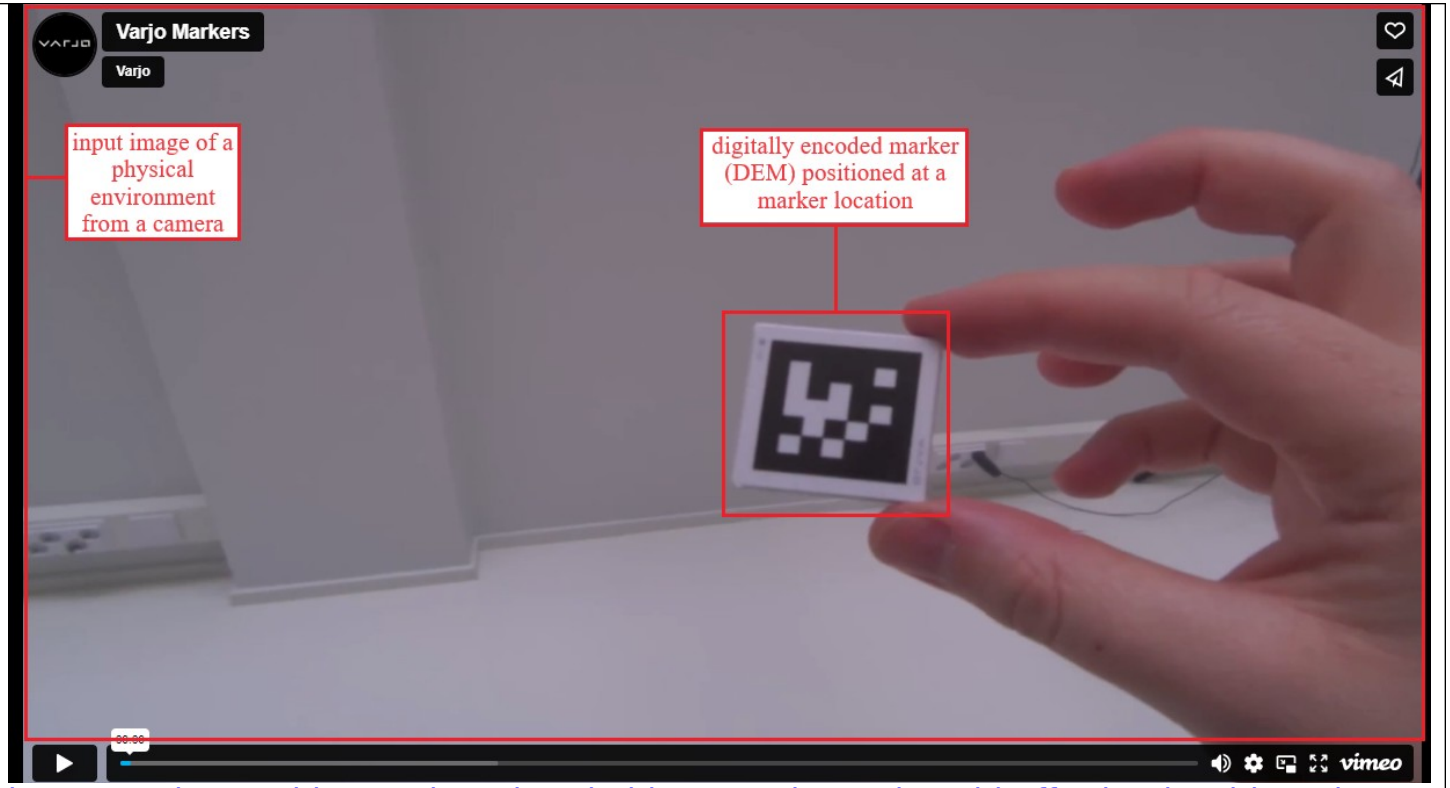
https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin

Now all you need is the actual logic of the marker manager. In this example script, we update the marker data in the *Update* loop. If Varjo Marker tracking is enabled, we get a list of the latest marker data and update all GameObjects in our *trackedObjects* array which have an ID matching that of a marker in the *markers* array. We also set the tracking mode of the marker here if required.

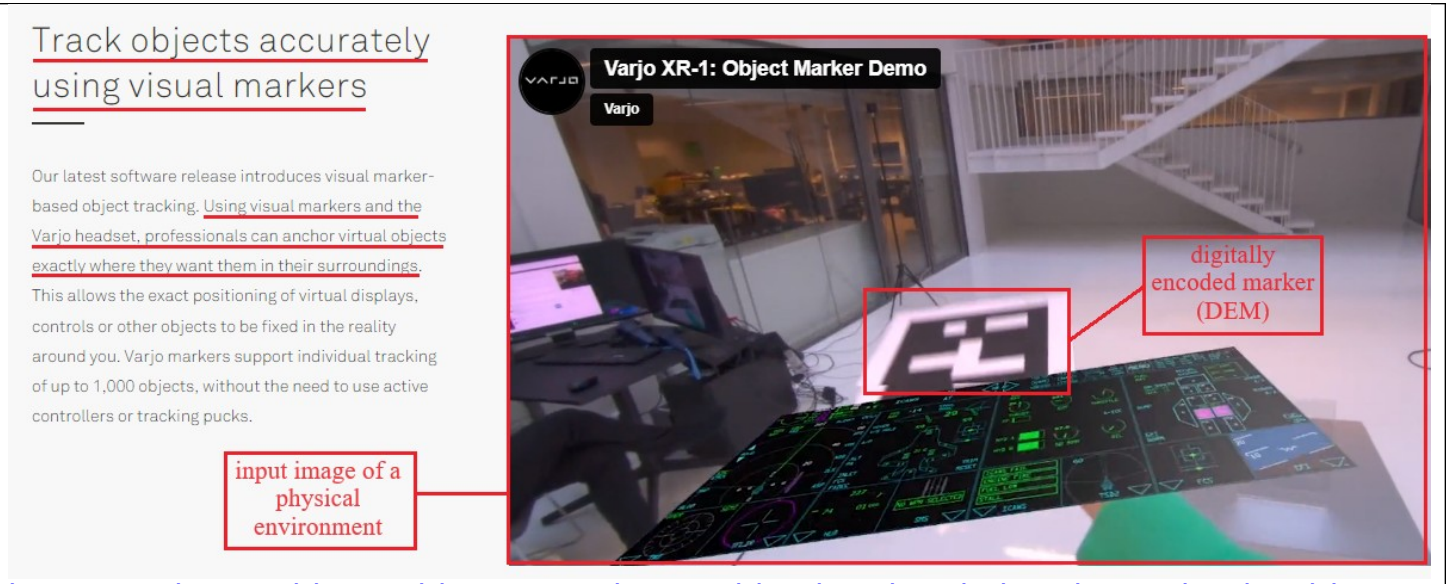
https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin

	Back in the Unity Editor, select the GameObject you want to follow the Varjo Marker. As the pose in the Varjo Marker is relative to the tracking space, you probably want to make the GameObject a sibling of the XR camera. Make sure that the GameObject is not flagged as static.  https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin
displaying an augmented reality image, on a display screen of the user device, including the input image and an overlay image representing the virtual object positioned within the	The accused product discloses displaying an augmented reality image (e.g., an AR image with an overlay of a virtual object), on a display screen (e.g., a display such as Bionic display of the Varjo XR-3) of the user device (e.g., an XR headset such as Varjo XR-3), including the input image (e.g., an input image of the physical environment captured by the pass-through camera of the Varjo XR-3 headset) and an overlay image (e.g., an overlay image of the virtual object) representing the virtual object positioned within the augmented reality image based on the decoded data (e.g., pose data for a Varjo Marker) from the DEM (e.g., a Varjo Marker) and the marker location (e.g., a location of the Varjo Marker). As shown below, an augmented reality image is displayed on the bionic display of the Varjo XR-3 headset. The augmented image contains the camera feed as well as the virtual object which is overlayed on the Varjo marker. The virtual object is mapped to

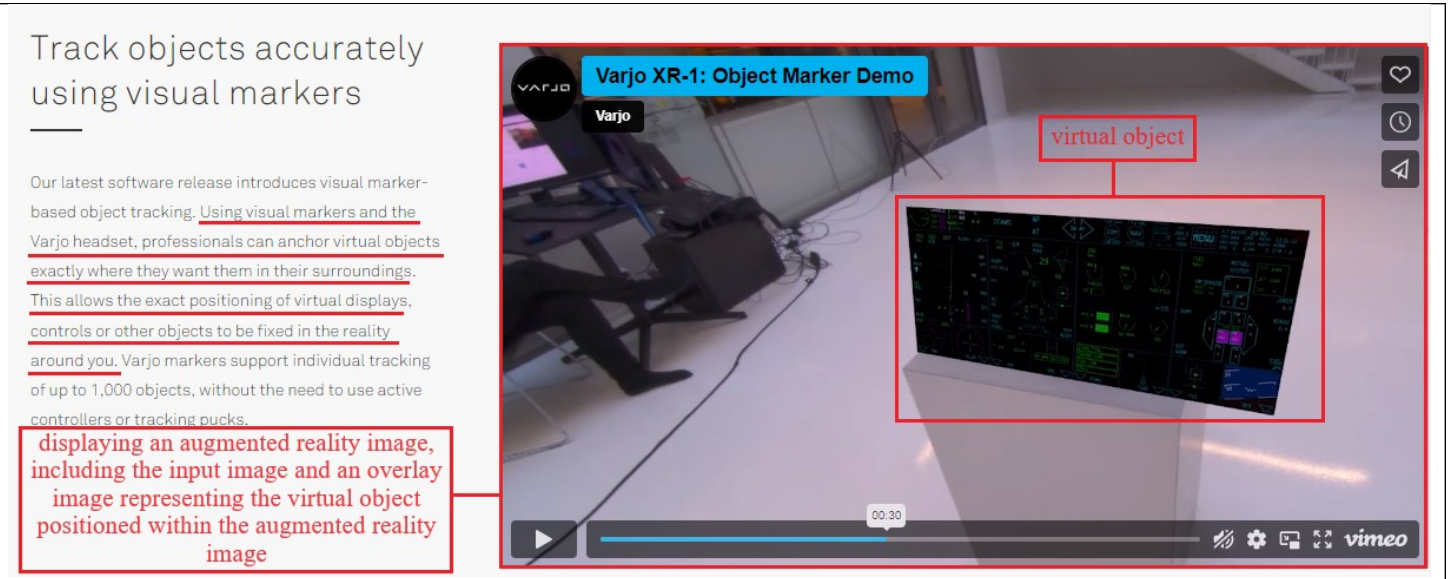
augmented reality image based on the decoded data from the DEM and the marker location,	the Varjo Marker using the actor in Unreal Engine, which uses the position data obtained from the Varjo Marker for positioning the virtual object.  https://varjo.com/products/varjo-xr-3/

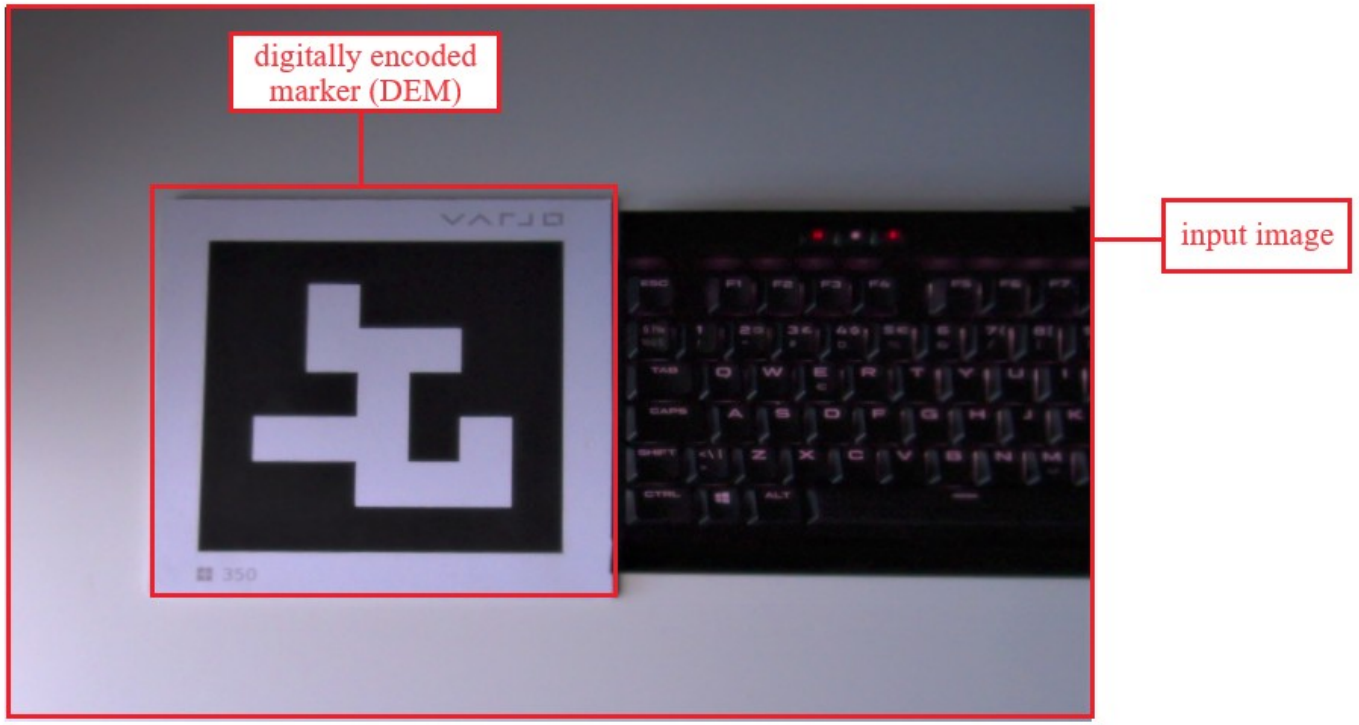
	 https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

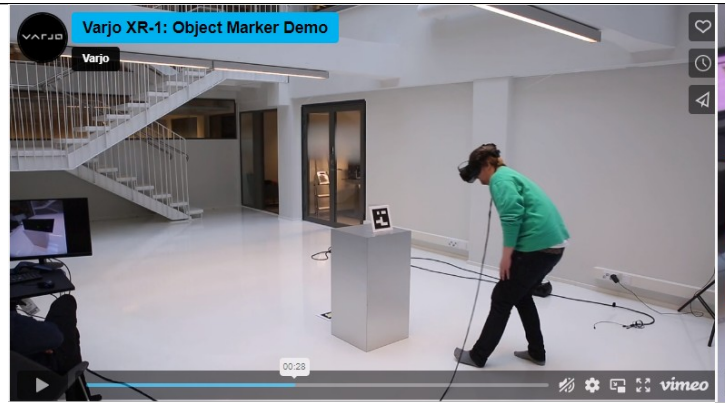
	 https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

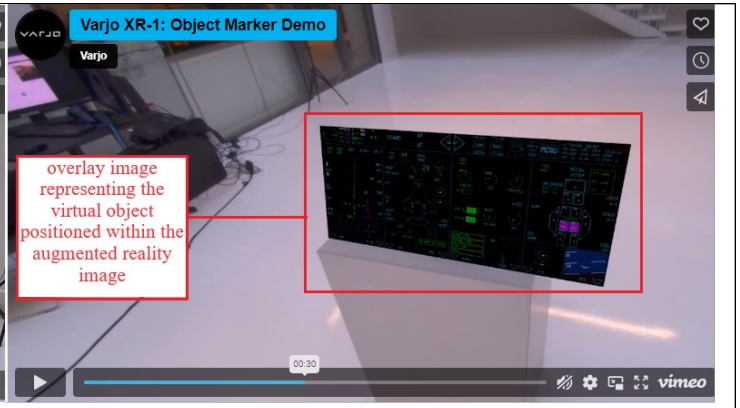


https://varjo.com/blog/webinar-recap-how-to-blend-real-and-virtual-seamlessly-with-new-xr-1-features/

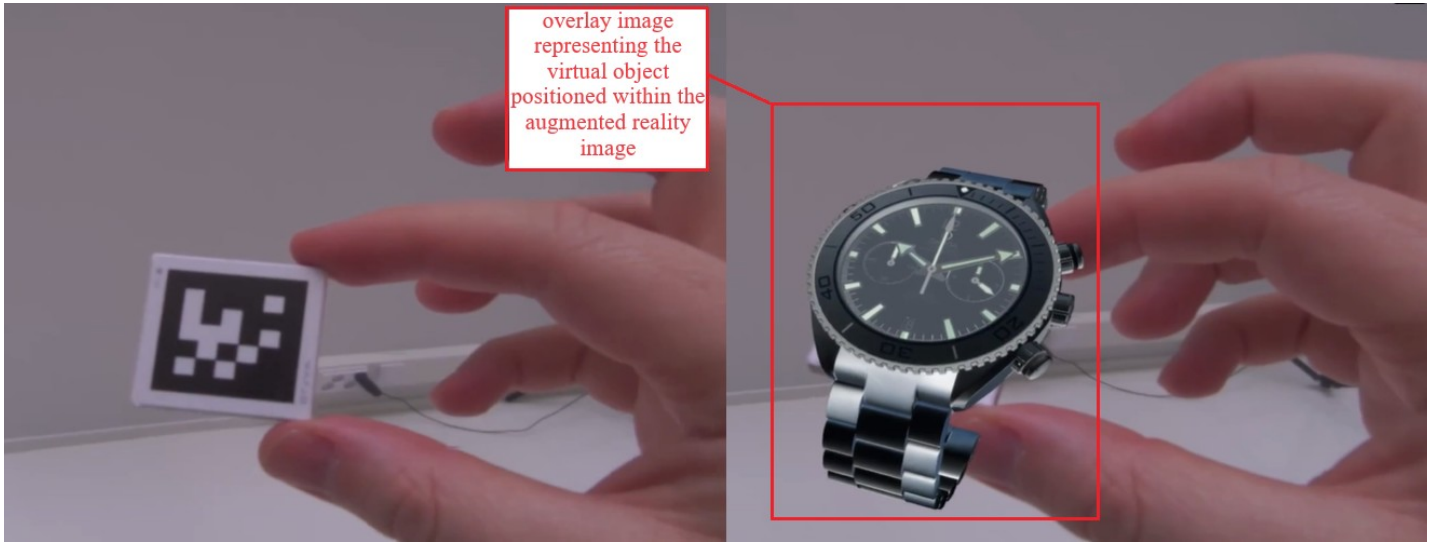
	 https://varjo.com/blog/webinar-recap-how-to-blend-real-and-virtual-seamlessly-with-new-xr-1-features/

	Place the printed marker next to your keyboard. Typically you would want to attach the marker to the physical object, so that they move together.  https://developer.varjo.com/docs/unity-xr-sdk/varjo-markers-with-varjo-xr-plugin

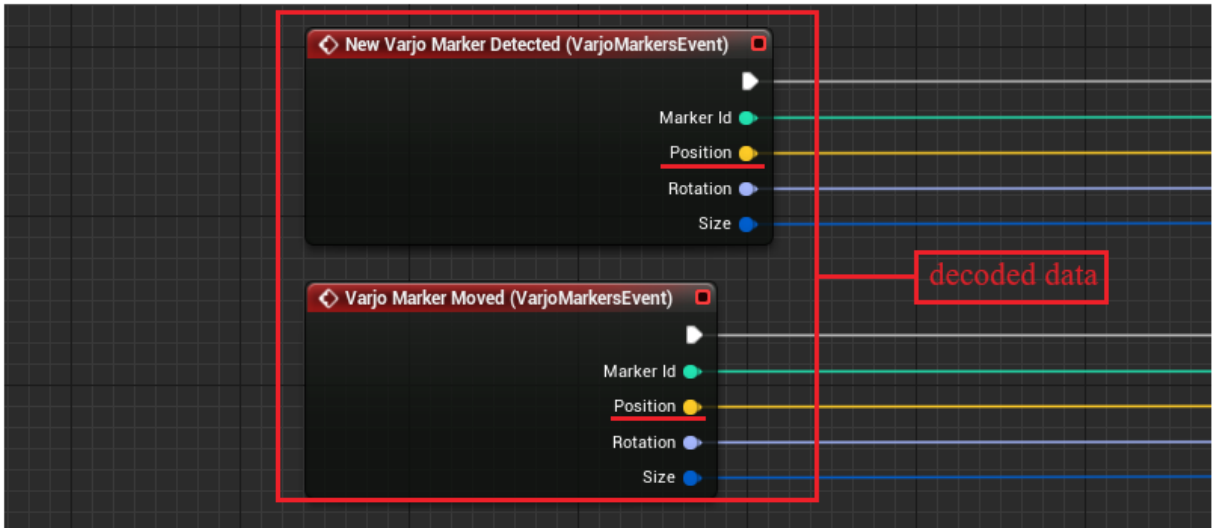




https://varjo.com/blog/webinar-recap-how-to-blend-real-and-virtual-seamlessly-with-new-xr-1-features/

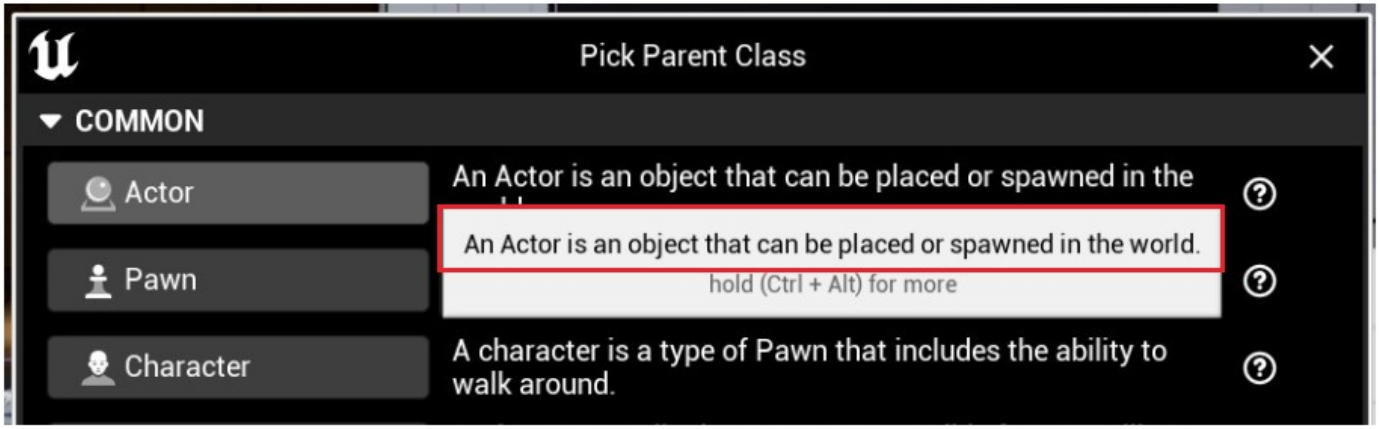


https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

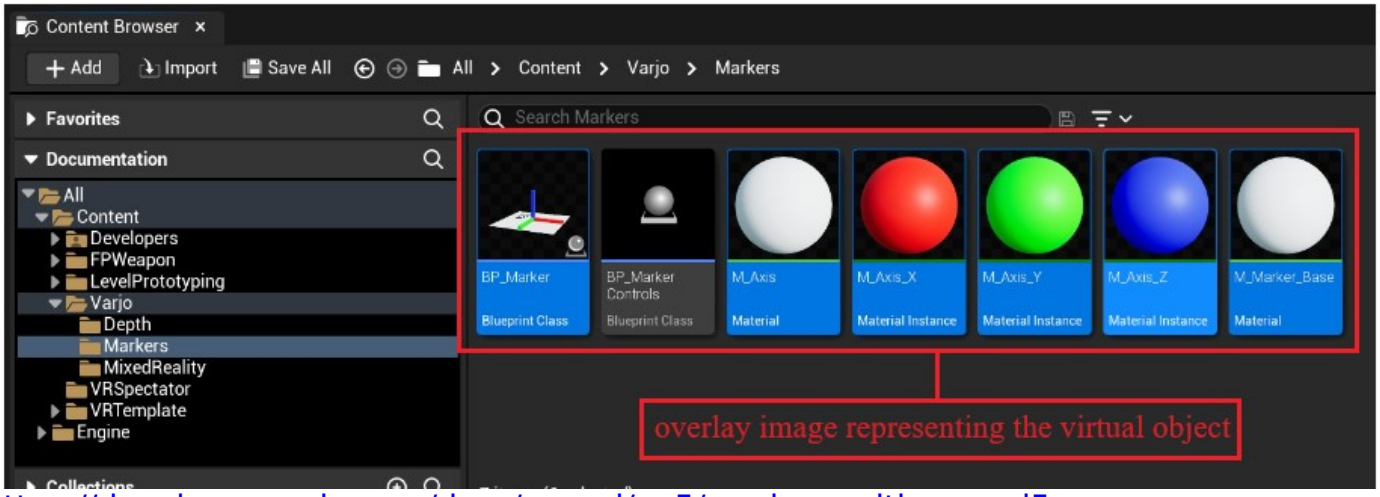
	All events provide a *Marker ID*, which matches the ID on the printed marker. In addition, **New Varjo Marker Detected** and **Varjo Marker Lost** events provide *Position*, *Rotation* and *Size* for the marker. The size of a marker never changes and it's provided in Unreal units, taking *World to Meters scale* into account so the size always matches the physical marker.  https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

Enable Varjo Marker tracking with **Set Varjo Marker Tracking Enabled**. Blueprint node **Is Varjo Marker Tracking Enabled** returns the current state.

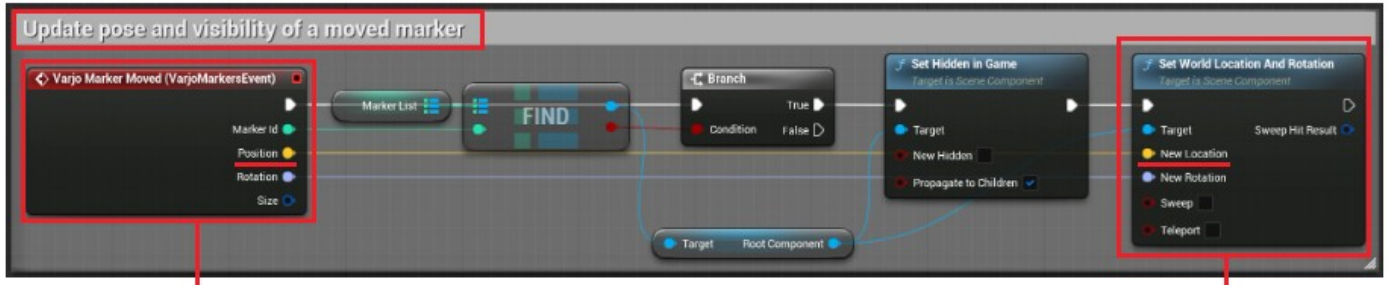
Create a logic for this. Follow the instructions below. First, create a new blueprint as **Actor** and name it **BP_MarkerControls**.

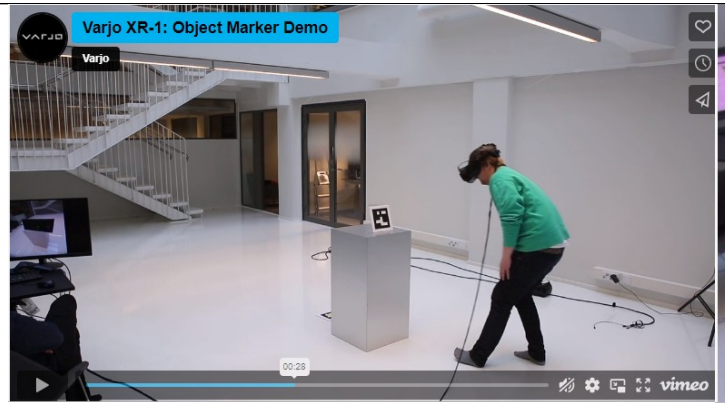


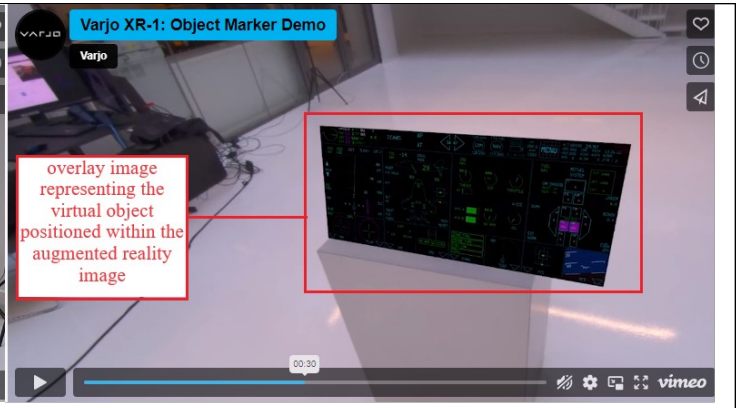
Create a new blueprint class **Actor** and name it **BP_Marker**.



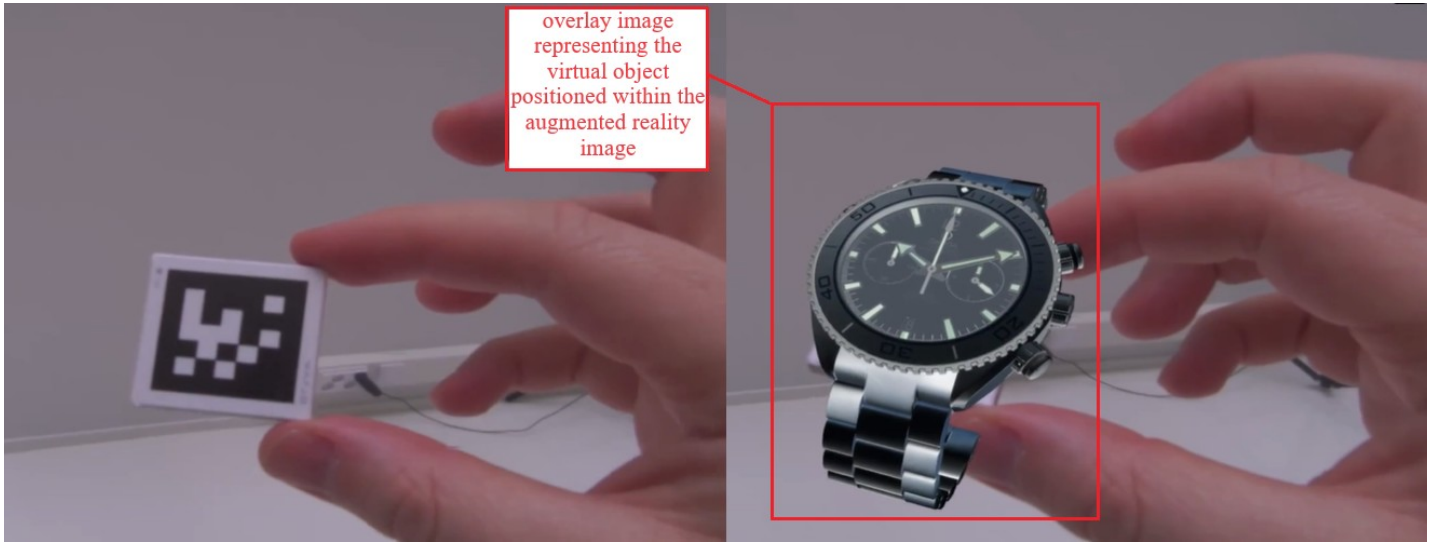
https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

	On **Varjo Marker Moved**, update the position and rotation of the Actor and ensure it is visible. The size of a marker never changes, so it is not necessary to update the scale of the Actor after it is created.  virtual object positioned within the augmented reality image based on the decoded data from the DEM and the marker location https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5
wherein the overlay image is positioned within the augmented reality image using the at least one of the geographic coordinate data and the relative coordinate data decoded from the DEM.	The accused product discloses wherein the overlay image (e.g., an overlay image of the virtual object) is positioned within the augmented reality image (e.g., an AR image with an overlay of a virtual object) using the at least one of the geographic coordinate data and the relative coordinate data (e.g., marker pose data) decoded from the DEM (e.g., a Varjo Marker). As shown below, the augmented image contains the camera feed as well as the virtual object which is overlayed on the Varjo marker. The virtual object is mapped to the Varjo Marker using the actor in Unreal Engine, which uses the position data obtained from the Varjo Marker for positioning the overlay image of the virtual object over the Varjo Marker.

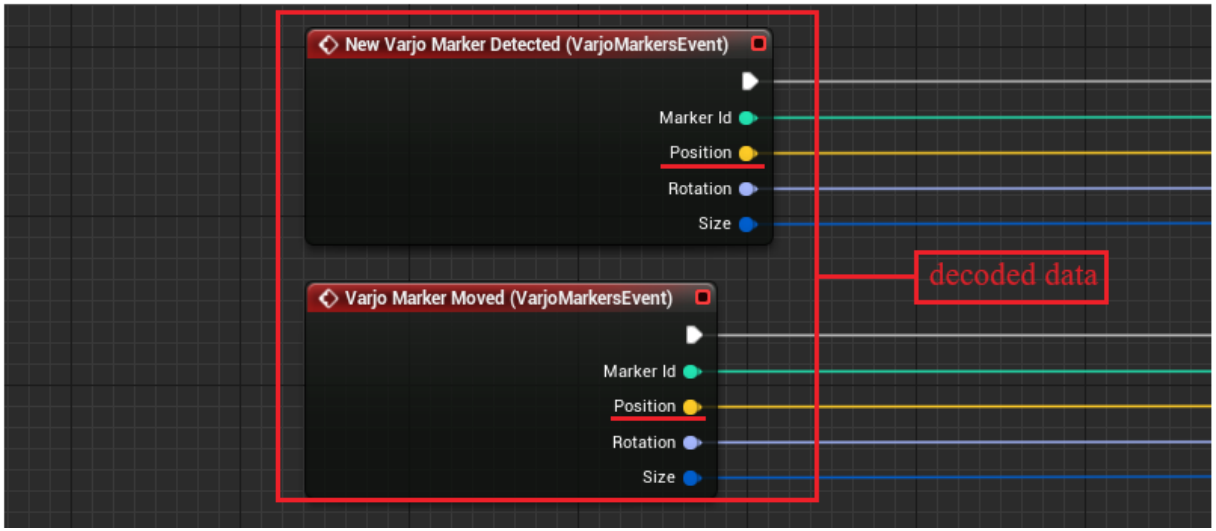




https://varjo.com/blog/webinar-recap-how-to-blend-real-and-virtual-seamlessly-with-new-xr-1-features/

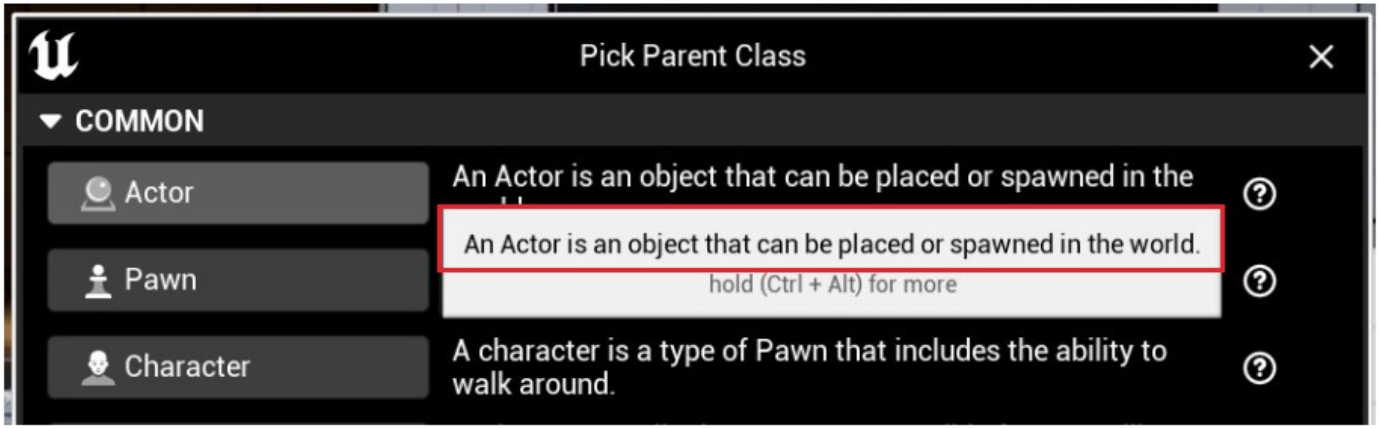


https://varjo.com/blog/anchor-virtual-objects-to-the-real-world-effortlessly-with-varjo-markers/

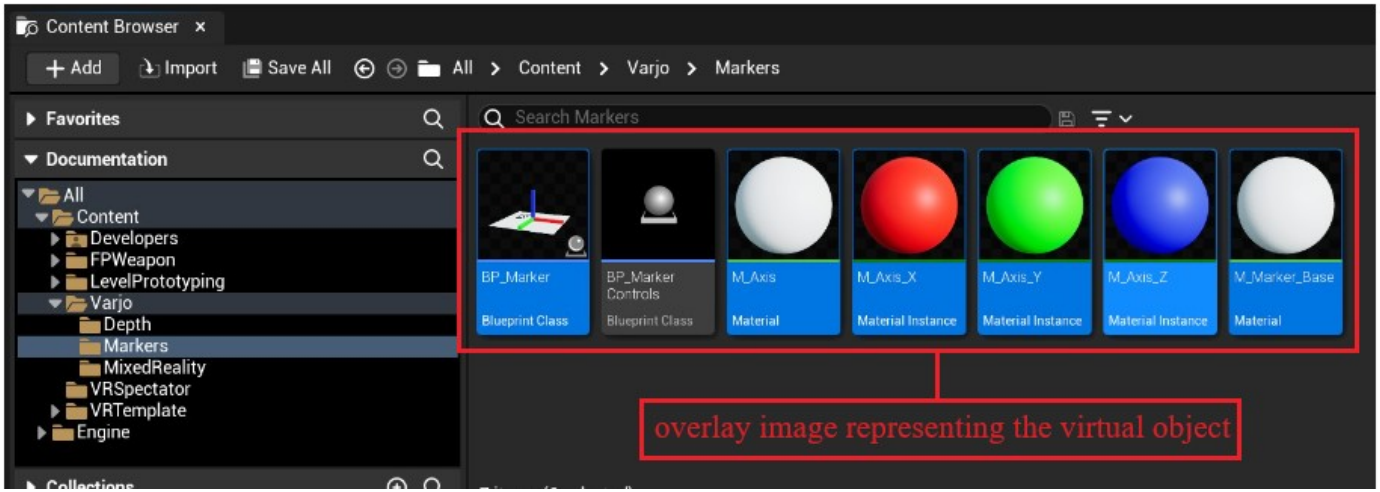
	All events provide a *Marker ID*, which matches the ID on the printed marker. In addition, **New Varjo Marker Detected** and **Varjo Marker Lost** events provide *Position*, *Rotation* and *Size* for the marker. The size of a marker never changes and it's provided in Unreal units, taking *World to Meters scale* into account so the size always matches the physical marker.  https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

Enable Varjo Marker tracking with **Set Varjo Marker Tracking Enabled**. Blueprint node **Is Varjo Marker Tracking Enabled** returns the current state.

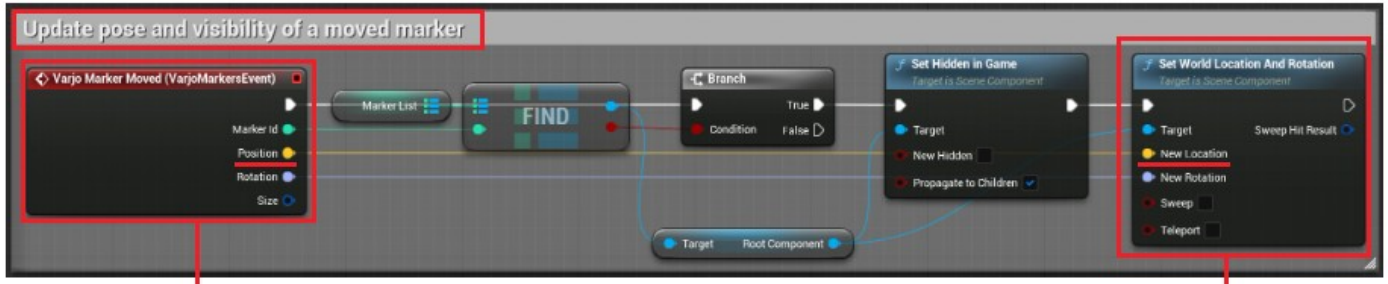
Create a logic for this. Follow the instructions below. First, create a new blueprint as **Actor** and name it **BP_MarkerControls**.

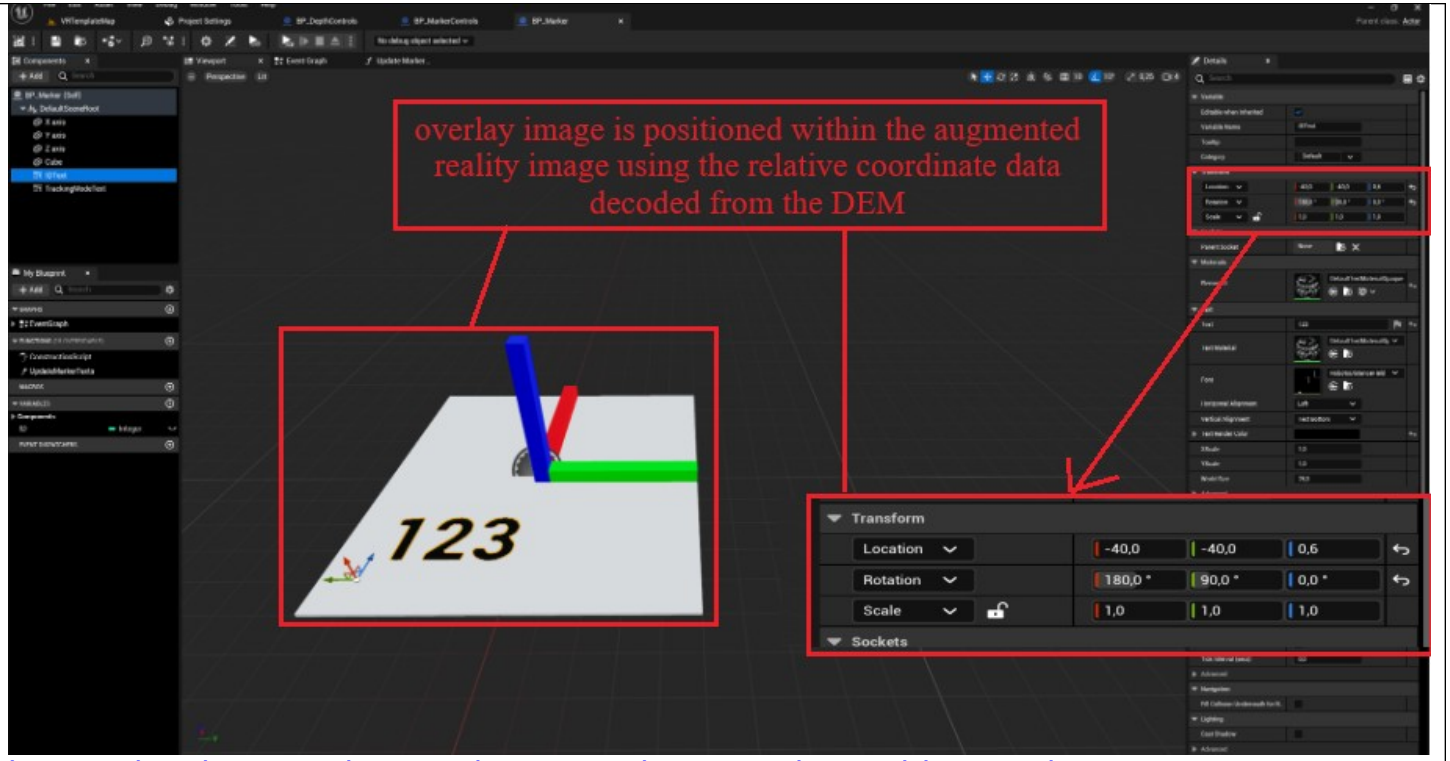


Create a new blueprint class **Actor** and name it **BP_Marker**.



https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

	On **Varjo Marker Moved**, update the position and rotation of the Actor and ensure it is visible. The size of a marker never changes, so it is not necessary to update the scale of the Actor after it is created.  virtual object positioned within the augmented reality image based on the decoded data from the DEM and the marker location https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5

	 https://developer.varjo.com/docs/unreal/ue5/markers-with-unreal5